UNIVERSITÄT
MANNHEIM

Scalable Propagation of Continuous Actions in Peer-to-Peer-based Massively Multiuser Virtual Environments: The Continuous Events Approach

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Wirtschaftswissenschaften
der Universität Mannheim

Vorgelegt von: Florian Heger

Mannheim, 2013

Dekan: Dr. Jürgen M. Schneider

Erstreferent: Prof. Dr. Christian Becker

Zweitreferent: Prof. Dr. Alexander Mädche

Tag der mündlichen Prüfung: 29.04.2013

Abstract

Peer-to-Peer-based Massively Multiuser Virtual Environments (P2P-MMVEs) provide a shared virtual environment for up to several thousand simultaneous users based on a peer-to-peer network. Users interact in the virtual environment by controlling virtual representations of themselves, so-called avatars. Their computers communicate with each other via a wide area network such as the Internet to provide the shared virtual environment. A crucial challenge for P2P-MMVEs is propagating state changes of objects in the virtual environment between a large number of user computers in a scalable way. Objects may change their state on one of the computers, e.g. their position. Information about a state change has to be propagated via the peer-to-peer network to computers of other users whose avatars are able to perceive the object. Optimization algorithms for a scalable propagation of state changes are needed because of the very large number of users and the typically limited bandwidth of their Internet connections. This thesis describes an approach that optimizes the propagation of state changes caused by continuous actions. Continuous actions lead to multiple subsequent state changes over a given period of time. Instead of propagating each subsequent state change caused by continuous actions via the network, the approach propagates descriptions of the actions included in so-called continuous events. Based on the descriptions, the subsequent state changes are calculated and applied over time on each user's computer. Continuous events contain information about (1) the timing of calculations, (2) the spatial extent of the influence of the continuous action in the virtual environment over time and (3) the effect of the continuous action on influenced objects over time. The propagation and management of continuous events is performed based on the spatial publish subscribe communication model. Each user computer declares interest in a certain space in the virtual environment. If the space intersects with the spatial extent of the influence of a continuous event, the particular computer is provided with the continuous event. This thesis describes the basic concept of continuous events, presents a system architecture for support of continuous events in the context of a given target system model for P2P-MMVEs, and evaluates the continuous events approach based on a prototypical implementation of the system architecture.

Acknowledgements

I would like to thank the following people and organizations for their support: My supervisor Prof. Dr. Christian Becker, Dr. Gregor Schiele, the project team of Peers@Play (especially Richard Süselbeck and Laura Krammer), Verena Majuntke, Sebastian VanSyckel, Dominik Schäfer, Christian Krupitzer, Markus Latz, Kerstin Goldner, the Mannheim Business School gGmbH, Prof. Dr. Dr. h.c. mult. Christian Homburg, Prof. Dr. Jens Wüstemann, Dr. Ingo Bayer, Björn Schenk, Angelika Hilger. A special thank goes to my parents, my brother and Susanne.

Contents

List of Figures

List of Tables

Nomenclature

AoE	Area of Effect
AoI	Area of Interest
CEC	Continuous Event Controller
CEE	Continuous Event Executor
CEM	Continuous Event Manager
CES	Continuous Event Storage Service
CR	Code Repository Service
CVE	Collaborative Virtual Environment
DARPA	Defense Advanced Research Projects Agency
DIS	Distributed Interactive Simulation
DL	Direct P2P Link Service
DVE	Distributed Virtual Environment
FoV	Field of View
HLA	High Level Architecture
IM	Interest Management
MASSIVE	Model, Architecture and System for Spatial Interaction in Virtual Environments
MMORPG	Massively Multiuser Online Roleplaying Game
MMVE	Massively Multiuser Virtual Environment
NPSNET	Naval Postgraduate School Networked Vehicle Simulator
NVE	Networked Virtual Environment
P2P-MMVE	Peer-to-Peer-based MMVE
PVS	Potentially Visible Sets
RWM	Random Waypoint Model
SIMNET	Simulator Networking Project
SPS	Spatial Publish Subscribe Service
TS	Timer Service
UFR	Update-Free Regions
VON	Voronoi-based Overlay Network

1. Introduction

Massively Multiuser Virtual Environment (MMVE) systems provide a shared virtual environment for a very large number of simultaneous users. These users are able to interact by controlling virtual representations of themselves via a software running on their computers. Communication in MMVE systems usually takes place over a wide area network, for example the Internet. MMVE systems originated from military simulations and became publicly known because of the media presence of the virtual online community Second Life and the commercial success of online roleplaying games such as World of Warcraft. Today, they are used in fields far beyond virtual communities or games. MMVE systems, for example, can be used as a cost-effective medium by companies for business-related use cases such as trainings of employees in virtual factory buildings, security simulations, virtual fairs and conferences.

Existing MMVE systems typically are provided by companies. Second Life, for example, is provided by Linden Labs. Using an existing MMVE system that is provided by another company can be a possible solution for certain business-related use cases. In the past, Second Life was used by companies for marketing purposes. Companies created virtual stores and interacted with their customers in the MMVE in order to build and maintain customer relations and to sell their goods. However, in order to be able to use an existing MMVE system, companies have to accept the economical and legal framework of the provider. This often involves strategic elements that are driven by the business model of the provider, for example the use of a given virtual currency. Accepting the conditions dictated by the provider is not necessarily beneficial for a company that wants to use the MMVE for their own business-related use case. In addition, existing MMVE systems are not flexible enough to support the requirements of a specific business-related use case of a company. For example, creating a virtual construction facility for specific training of employees is hard to realize in an existing MMVE system. In order to control the training of employees and to measure their performance, companies need sophisticated evaluation mechanisms that have to be tailored to the use case and given goals. This requires adjustments to the software of the MMVE system and usually

can not be realized based on an existing MMVE system.

In conclusion, creating and running an own MMVE system tailored to the specific use case seems to be the best solution for companies and their business-related use cases. However, creating and running an MMVE system is very expensive. In practice, existing state-of-the-art MMVE systems are based on a client/server architecture. This type of architecture is well understood by MMVE system designers and, in practice, works well. The MMVE providers, for example the Second Life provider Linden Labs, maintain huge server farms that run their MMVE. The computers of the MMVE users connect to the servers as clients. All communication between the clients is managed by the central servers. The scalability of the overall MMVE system is usually enhanced by splitting the virtual environment into smaller regions and assigning the management for regions to servers in a server cluster. A Second Life region server, for example, is responsible for managing a virtual region with a size of 256 x 256 meters and is able to support up to hundred simultaneous users within its region. In addition, the scalability of the system is enhanced by increasing the hardware resources of the server clusters. As a result, MMVE providers face huge costs for the provision of large computer centers. In an interview given in 2006, the chief technology officer of Linden Labs explained that Second Life at that time was run by 2597 servers [Ter]. Therefore, providing an own MMVE based on a state-of-the-art MMVE system with a client/server architecture would result in huge costs for companies and is not a feasible solution.

In past research, it was shown that a peer-to-peer architecture [LCP+05, ATS04] can be an alternative [KMA02, KLXH04, HL04, YV05, FRP+08] or addition [IHK04, CYB+07, SZ08, BHS+08, Kul09] to client/server-based MMVE system architectures. In peer-to-peer architectures, user computers act as peers. Communication flows directly between the user computers and tasks for running the system are assigned to user computers. Using a peer-to-peer architecture for the provision of MMVEs allows to run the system mostly on the computers of the participating users. Own hardware by the MMVE provider is only needed for certain tasks such as the initial bootstrapping of the system and the persistent storage of the virtual environment. This has the potential to make MMVEs based on peer-to-peer cheaper in comparison to client/server-based MMVEs. In conclusion, providing an own MMVE based on a peer-to-peer architecture would result in less costs than using a client/server architecture and is a more beneficial solution for companies.

The distributed character of a peer-to-peer architecture results in a lot of new chal-

lenges for the design of the MMVE system. Instead of sending informations over the network to one central server cluster, user computers have to send informations over the network to a potentially large number of other user computers. In addition, system tasks such as controlling non-user characters and environmental effects have to be assigned to certain user computers and coordinated between the user computers by sending informations over the network. The increased amount of direct information flow between the user computers can lead to problems for the scalability of the overall MMVE system because standard network connections of users usually have a very limited upload bandwidth. Therefore, a huge challenge for the design of a scalable *peer-to-peer-based MMVE* (P2P-MMVE) system is to minimize the information flow that has to be sent by peers over the network.

In order to make a P2P-MMVE system scalable, numerous specific approaches and algorithms are needed. The continuous events approach, which is presented in this work, focuses on one aspect of scalability in P2P-MMVE systems. It aims at providing a scalable propagation of continuous actions. Continuous actions are actions that result in numerous following state changes over time. Continuous actions are described in more detail later on in this work. In the following, the research objectives of this work are elaborated. After that, a summary of the approach is given and the main contributions are identified. Finally, an overview of the organization of this work is given.

1.1. Research Objectives

P2P-MMVEs include so-called *continuous actions*. A continuous action is an action or user input that results in multiple following state changes of objects in the virtual environment over time. A relatively simple example can be described as follows: As result of a user input, an object in the virtual environment moves a certain distance with a given starting and a given ending location. After it arrived at the ending location, it influences other objects within a certain distance. A more complex example for a continuous action that influences multiple objects can be described as follows: A peer is assigned the task of controlling the weather of the virtual environment. As part of this task, it has to create and control a large number of rain cloud objects. Each of these cloud objects has to be created, displayed and moved on other peers. After the initial creation of a new rain cloud on a peer, the cloud changes its position numerous times according to the wind. On its way, the

rain cloud influences numerous other objects, for example because rain falls down on the objects. Without any optimization, in these examples each position change of the object and each state change of other objects caused by the initial object has to be represented by an update that has to be sent over the network to other peers. This results in a potentially huge amount of network traffic.

The network connections of users typically have a much lower upload bandwidth than download bandwidth. This is in general a potential bottleneck in a P2P-MMVE system. Sending the potentially large number of updates for continuous actions without further optimization over the network even increases the danger of bottlenecks at the sending side of the network connections. Therefore, finding an approach for optimizing the sending of the outcome of continuous actions over the network is crucial for the overall scalability of a P2P-MMVE system.

This work explores ways to optimize the sending of the outcome of continuous actions over the network in P2P-MMVEs. It aims at finding an approach to *send the outcome of continuous actions to all affected peers with as little network messages and bandwidth as possible*. For this work, a target P2P-MMVE system model is given. This work puts an emphasis on the *design of a system architecture and algorithms for sending the future outcome of continuous actions in the context of the target system* in a scalable way.

1.2. Overview of the Approach

The presented approach builds on the basic idea to send *descriptive informations* about continuous actions over the network instead of sending an update for every single state change that is caused by a continuous action over time. The approach introduces an explicit event entity for continuous actions, so-called *continuous events*. Continuous events are used within the P2P-MMVE system to carry the descriptive informations about continuous actions. After its creation, a continuous event is propagated to all affected peers. In the following, the resulting state changes of the continuous action that is described by the continuous event are calculated over time, at best without having to send any further messages over the network.

Continuous events include descriptive informations about timing, the spatial influence over time and the effect over time of the described continuous action. The informations about *timing* include the points in time when the continuous action

starts and ends and the interval between the calculation of new updates on the target peers. The informations about the *spatial influence over time* include the initial influence area of the action and how this area changes over time. The informations about the *effect* describe the specific state change that should be applied to affected objects in the virtual environment.

Instead of including static informations, continuous events carry references to locally stored *function code*. In order to determine the spatial influence over time and the effect at a given point in time and to apply the effect to all affected objects, a continuous event only includes parameter values and calls the function code according to the references. Function code is supported in a generic way. Additional code can be added to the system and called by continuous events.

Continuous events are *executed and managed automatically* by the system. A continuous event can be started on one of the peers and then is propagated to other peers based on the spatial influence of the continuous action. All target peers execute continuous events automatically. The execution of continuous events includes the calculation and application of the resulting state changes over time. Selected peers are assigned a management role. These peers are responsible for management tasks, such as providing new peers with existing continuous events and coordinating the potential modification and termination of existing continuous events.

The approach is integrated into the target P2P-MMVE system model by using *artificially created single events*. When executing continuous events, the calculated state changes are not applied directly to objects. Instead, artificial single event messages are created that are identical to event messages received from the network. This allows other parts of the target P2P-MMVE system to perform their algorithms without further adjustments.

1.3. Contributions

The contributions of the continuous events approach to the field of P2P-MMVEs can be summarized as follows:

Enhancement of scalability of state change propagation in P2P-MMVEs - The approach allows to propagate all future state changes that are caused by a continuous action in an aggregated way by using one continuous event. This results in a reduction of network traffic for propagating the future outcome of continuous actions and enhances the overall scalability of state change propagation in P2P-MMVEs.

Support of generic types of complex continuous actions - Existing optimization approaches for propagation of continuous actions typically focus on a certain type of continuous action. The continuous events approach aims at supporting generic function code to describe the future outcome of continuous actions. This enables the support of a wide variety of action types and use cases. In addition, a continuous event can include informations about very complex state changes of multiple objects. After a continuous event was received by a peer, the approach is able to calculate and apply state changes of a potentially large number of objects on this peer.

Encapsulated execution and management of continuous events - The design of the continuous events approach facilitates the use of continuous events by giving interfaces for the explicit start, modification and termination of continuous events. The following processes for propagating continuous events to affected peers, calculating continuous events and applying state changes are by design encapsulated in the system for continuous event support. Once a continuous event was started on a peer, the system automatically takes care of distributing continuous events to peers, calculating and applying state changes on these peers and providing new peers with existing continuous events. In case a modification or termination of an existing continuous event was performed via the given interfaces on a peer, the system for continuous event support automatically takes care of the distribution of this information to other peers.

1.4. Organization of this Work

In addition to this introductive chapter, this work is organized as follows:

Chapter 2 describes the background of this work. An introduction into MMVEs is given and MMVE systems from research and practice are presented. An overview of related work from the fields of interest management, dead reckoning, prediction techniques for avatar behavior and geocast is given. Previously published own related work is summarized. The target system model of this work is described. The underlying assumptions and requirements for this work are presented.

Chapter 3 presents the continuous events approach for a scalable propagation of continuous actions in P2P-MMVE systems. The concept of continuous events is introduced. A formal model of continuous events is given. Timing alternatives for continuous events are discussed. Potential use cases for very complex continuous events are described and discussed.

Chapter 4 describes the design of a system architecture for continuous event support in the context of the given target system model for P2P-MMVEs. The overall design and architectural components are presented and the integration into the target system model is explained. The system behavior and algorithms for basic continuous event support are given. Several extensions to the basic continuous event support are discussed.

Chapter 5 includes an evaluation of the continuous events approach. The evaluation focuses on the networking aspects of the approach. Simulations of four varying types of continuous actions are described. The performances of two alternatives of the continuous events approach with varying spatial modelings are compared to the performance of a P2P-MMVE without continuous events and the performance of a state-of-the-art MMVE architecture based on the client/server model and dead reckoning.

Chapter 6 finally draws conclusions from the presented work. A comprehensive summary of this work is given. This work is classified in comparison to related work. The fulfillment of requirements by the continuous events approach is assessed. Potential fields of future research are identified and discussed.

2. Background

This chapter describes the background of this work. Section 2.1 gives an introduction into MMVE systems in research and practice. Section 2.2 presents an overview of related approaches and technologies from the field of MMVEs and other research fields. Section 2.3 describes the target system model of this work. Section 2.4 presents the underlying assumptions. Finally, Section 2.5 presents the requirements for this work.

2.1. Massively Multiuser Virtual Environments

A *Massively Multiuser Virtual Environment* (MMVE) is a large shared virtual environment for a huge number of simultaneous users. The MMVE is displayed to the users via software on their computers. The users can interact with each other or the environment via virtual representations, so-called *avatars*. The user avatars are typically controlled by user inputs into peripheral devices such as keyboard and mouse. The software translates the inputs into avatar actions. Communication in a MMVE system typically takes place over a wide area network such as the Internet. Depending on the basic communication model of the MMVE system, the provision and management of the MMVE can either be done centralized by using large server clusters or decentralized by distributing the tasks to user computers.

Over the last decades, multiple research communities and military initiatives evolved around the idea of providing a shared virtual environment for a huge number of users over wide area networks. Shared virtual environments with a focus on entertainment like World of Warcraft became a major factor for the gaming industry. Virtual communities like Second Life gained huge attention by the mass media. In the following, this section gives an overview of MMVE systems in research and practice. The overview focuses on introducing the varying research communities, military initiatives and commercial MMVE systems and only includes short descriptions. The following section then gives more detailed descriptions of approaches and technologies that are directly related to this work.

The U.S. military founded several projects with the goal to support distributed military simulations over wide area networks, starting with the *Simulator Networking project* (SIMNET) [MT95]. SIMNET was developed between 1983 und 1990 and sponsored by the Defense Advanced Research Projects Agency (DARPA). The project aimed at providing an infrastructure for military training in a virtual environment.

Later on, SIMNET research evolved into the research field of *Distributed Interactive Simulation* (DIS). The DIS initiative was driven by the U.S. government and industry. DIS built on research work from SIMNET and aimed at defining a generic infrastructure for distributed simulations in virtual environments. The DIS infrastructure, which resulted from the initiative, provides a shared virtual environment based on distributed autonomous simulation nodes. It is able to simulate a large number of military units such as tanks and air planes. The DIS infrastructure is described by several IEEE standards [iee93, iee95a, iee95b, iee96, iee97, iee98].

The *High Level Architecture* (HLA) was another research initiative driven by the U.S. government. It used conceptual elements from the DIS standards and added concepts from other previous military projects in the field of distributed simulation, for example the aggregate level simulation protocol [WW94]. The HLA initiative aimed at providing a general framework for virtual environment systems and simulations. The HLA framework, which resulted from the initiative, allows developers to structure and design their simulation applications in a generic way without emphasizing certain use cases or application types. Analogical to DIS, the HLA is described by several IEEE standards [iee03, iee07, iee10a, iee10b, iee10c].

Collaborative Virtual Environment (CVE) research focused on the collaboration aspect of virtual environments. It aimed at providing 3-dimensional virtual environments that primarily support collaborative work and social play [BGRP01]. CVE systems with importance for this work are *NPSNET* and *MASSIVE*. Both systems are introduced in the following. For further informations about other CVE systems and a comprehensive overview of CVE research see [BGRP01].

The *Naval Postgraduate School Networked Vehicle Simulator* (NPSNET) put a strong emphasis on the support of a very large number of users. It incorporated existing technologies, for example the SIMNET database technology and the DIS communication protocol, and aimed at enhancing their capabilities based on multicast networks in order to be able to support more than 1000 simultaneous users over the Internet in a scalable way. [MBZ+95]

The *Model, Architecture and System for Spatial Interaction in Virtual Environments* (MASSIVE) originated from the University of Nottingham, UK [Gre97]. Several prototypes were developed over the years. MASSIVE-1 focused on teleconferencing over wide area networks. MASSIVE-2 and MASSIVE-3 focused on a wider variety of use cases such as public participation in online art and performance and inhabited television. Inhabited television can be described as a combination of CVEs and television that aims at supporting television shows in a virtual environment with public participation via avatars. [BGRP01]

A new genre of games, *Massively Multiuser Online Roleplaying Games* (MMORPGs), became very successful in the late 1990s and early 2000s and evolved into a mass phenomenon. The most successful MMORPG, *World of Warcraft* [Blia], had a peak number of worldwide subscribers of more than 12 million in 2010 [Blib]. Today, MMORPGs are a major part of the gaming market and every year multiple new MMORPGs are published. The concepts from these systems are highly relevant for this work. Unfortunately, the gaming industry does not publish their concepts for reasons of competition. Therefore, related work from MMORPG systems can only be regarded by this work at best knowledge.

During the 2000s, shared virtual environments with a focus on community aspects gained large public attention, mainly because of the media buzz caused by the system *Second Life* [Lina]. Second Life aimed at providing a virtual universe where users can live a parallel life. Similar to the real world, users are able to buy and occupy real estates and goods. Users are involved in forming the virtual world, for example they are able to add own content to the virtual environment. Second Life even included an own currency and offered the possibility to exchange real money into virtual money. Driven by the strong media presence, numerous companies established virtual representations and stores within Second Life. In contrast to MMORPGs, more information is available about the technical background of Second Life, for example via the Second Life Wiki [Linb]. The source code of the Second Life client software was even published under Gnu Public License (GPL) in 2007.

2007 also marked the year of founding of the *OpenSimulator* project [Ope]. The project followed up the GPL publication of the Second Life client software. It initially aimed at providing an alternative open source MMVE server that could be used in conjunction with the Second Life client. Although the compatibility with the Second Life client is still maintained today, the aim of the project shifted over time. At the point in time of the writing of this work, OpenSimulator supports a variety

of client softwares and the focus of the project is on providing a server software that can function as an infrastructural backbone for a 3-dimensional version of the World Wide Web (a so-called *3D Web*). To date, the OpenSimulator community is very active and there is also involvement by industry partners, mainly by IBM. IBM even introduced a product for virtual collaboration based on OpenSimulator in 2009: Virtual Collaboration for Lotus Sametime.

Although the topics of distributed communication and provision of shared virtual environments were discussed in research and provided by some of the military simulations, commercial MMVE systems were and today still are using the client/server communication model. This model gives the companies that run the MMVEs more control over the system architecture, but comes with high costs for running huge server farms. In the 2000s, researchers proposed peer-to-peer as an alternative communication model for MMVEs. They aimed at developing systems that run the MMVE either fully or at least partly without central server hardware and are able to reduce the costs for running MMVEs. Several research communities evolved around the idea of developing MMVEs based on peer-to-peer. These communities initially proposed such *peer-to-peer-based MMVE* (P2P-MMVE) systems under the terms *Distributed Virtual Environment* (DVE) and *Networked Virtual Environment* (NVE). Over time, the research communities joined forces and agreed to MMVE as the general term in order to accentuate the aspect that these systems aim at very large user numbers [SHWL09]. Numerous proposals for P2P-MMVE systems were made that are highly relevant for this work because it aims at P2P-MMVEs. The relevant systems from the field of P2P-MMVEs and their approaches are described in detail as part of the following section.

2.2. Related Concepts and Technologies

This section gives an overview of related approaches and technologies from MMVE research and practice as well as other research fields. The presented approaches and technologies are categorized based on the general concepts of interest management, dead reckoning and prediction techniques for avatar behavior. In addition, this section describes the concept of geocast from the research field of context-aware computing. Finally, it gives a summary of previously published own related work.

2.2.1. Interest Management

MMVEs typically consist of a huge number of stateful objects. In order to display an up-to-date state of these objects on all participating user computers, a large number of state update messages has to be propagated between the computers. Because of the huge number of objects and the large size of the environment in a MMVE, a system-wide propagation of all state updates to all computers poses a major challenge for the scalability of the system and usually does not scale at all.

Therefore, MMVE systems enhance scalability by performing filtering algorithms before the propagation of state updates. The relevancy of state updates for certain users is determined based on one relevancy criterion or multiple revelancy criteria and only the relevant state updates are propagated to each user. In other words: The filtering algorithms determine the *interest* of a user in certain update messages. By reducing the message amount for update propagation down to only the relevant messages, these so-called *interest management* algorithms enhance the overall scalability of a MMVE system.

To date, interest management algorithms are a crucial part of every MMVE system. This work also uses basic interest management concepts such as the area of interest and a determination of relevancy of events based on spatial areas. Therefore, interest management approaches and technologies from research and practice are highly relevant in the context of this work. In the following, this work gives an overview.

Over the years, numerous interest management approaches were published and a variety of possible relevancy criteria were proposed. In the following, the related work about interest management is classified based on the interest criterion that is used by the particular approach. The following categories of interest management are presented: Aura-based interest management, extended aura-based interest management, zone-based interest management, interest management based on visibility relations, and interest management based on content attributes. Zone-based interest management is further classified into approaches that use a static tesselation into zones and approaches that perform a dynamic tesselation at runtime. Figure 2.1 on page 14 gives an overview of the classification structure.

Please note that the presented related work focuses on interest management approaches for distributed system architectures. For a comprehensive overview of interest management approaches for centralized client/server-based system architectures with a focus on massively multiplayer games see [BKV06].

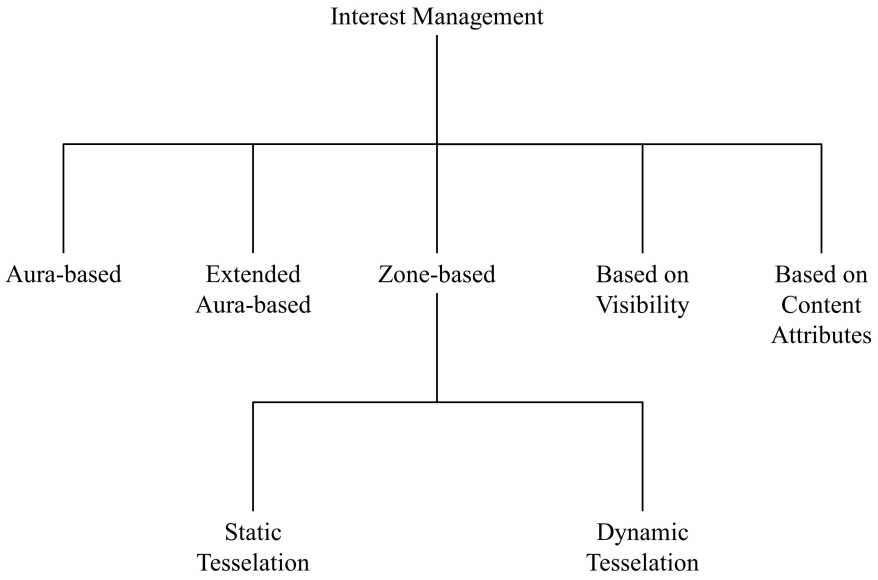


Figure 2.1.: Classification of interest management approaches

Aura-based Interest Management

The conceptual foundation of *aura-based interest management* is the *aura model*. The aura model is a concept from the *spatial model of group interaction in virtual environments* by Benford et al. [BF93]. The spatial model of group interaction originated from CVE research and was used, for example, by the CVE system MASSIVE [Gre97].

In the aura model, every object in the virtual environment is surrounded by a spatial area, the so-called *aura*. Objects carry their auras with them in case the object position changes. The system that provides the infrastructure for the virtual environment watches for overlapping object auras. An interaction between objects is possible if the corresponding auras of the objects overlap. Objects can have multiple auras representing different media, for example different auras for visibility and for audio. These auras can have varying spatial structures. This allows asymmetrical interaction. In a large space it is possible that a user can see another user, but at the same time the user is not able to hear the other user. This scenario, for example, can be modeled by using a very large visibility aura and a much smaller audio aura. The avatar of the user that can be seen and not heard is located within the large visibility aura, but outside of the smaller audio aura. The authors further

divide auras into *focus* and *nimbus*. A focus aura describes the potential perception of a user avatar within the virtual environment, for example the viewing distance. A nimbus aura describes the potential perceptivity of an object. A user is able to perceive an object if the focus aura of the user intersects the nimbus aura of the object. [BF93, GB95]

Today, the aura model or at least a modified version of the aura model is part of most MMVE systems. The interest of a user avatar in its surroundings is typically modeled by an aura that surrounds the avatar, the so-called *area of interest* (AoI). The corresponding user computer is provided with all updates about state changes of objects that take place within the avatar's AoI. Depending on the particular MMVE system, this basic concept is further extended by additional concepts. In the following, this work gives an overview of aura-based interest management approaches.

The P2P-MMVE system *HyperVerse*, which was described by Botev et al., includes an interest management approach based on multiple 3-dimensional auras. The system uses three spheres that surround every user avatar. The avatar's *field of view* (FoV), the AoI, and a third sphere without a particular name. The FoV has a radius d and depicts the view of the user on the environment. The AoI has a much larger radius and is used to retrieve objects and terrain. Because the FoV is much smaller than the AoI, the user can move a certain distance without additional retrieval of object and terrain data. The third sphere with a radius between d and the radius of the AoI is used to reset the AoI position. In case the FoV sphere exceeds the third sphere, the AoI is reset and a new retrieval of object and terrain data is performed. [BHS+08]

Kawahara et al. proposed a *peer-to-peer-based message exchange scheme* for large-scale NVEs. Their approach surrounds the avatar of each peer with an AoI. All peers establish unicast connections to other peers whose avatars are located within the AoI of the peer (so-called *active entities*). By establishing these connections, an overlay network for propagation of update messages is created. A peer provides its active entities with detailed update messages about itself via the unicast connections. In addition, it compresses the minimum required information about all of its active entities and exchanges this information with other active entities. By doing so, these active entities are provided with rough information about avatars that are located close by but not within their own AoI (so-called *latent entities*). The overlay network adapts dynamically. Existing unicast connections are disconnected and new

connections are established according to avatar movement. [KMA02]

The CVE system *VELVET*, which was proposed by de Oliveira et al., includes an interest management approach with dynamically adjusted AoI sizes and AoI border areas for object prefetching. In VELVET, every user avatar is surrounded by a 2-dimensional circular AoI. The AoI size of each avatar can vary and is dynamically enlarged or shrinked at runtime based on the number of objects that are located within the AoI. In case of a high object density within the AoI, the AoI is shrinked in order to reduce the number of received update messages and to maintain system scalability. In case of a low object density within the AoI, the AoI is enlarged in order to optimize the utilization of the user hardware and network connection. Because of the potentially varying AoI sizes, the interest between two users is not necessarily symmetric. For example, a user A can have a larger AoI than another user B. As a result of this, user A can see user B while user B can not see user A. The authors call this 'degree of blindness' [dOG02, p. 2493] and define this as 'perfectly legal in VELVET' [dOG02, p. 2493] because they only aim at providing consistency at best effort. In addition to the dynamically adjusted AoI sizes, the AoI scheme of VELVET includes a border area for each AoI. The border area is used to perform check-in and check-out operations for objects. Based on the border area, object data can be prefetched before the location of an object shifts within a user's AoI because of user movement. The border area can also be dynamically adjusted. In VELVET, updates are propagated via multicast. A multicast group is created for each object. Users join the multicast groups of all objects that are located within their AoI. In case the state of an object changes, the object sends an update message to its multicast group. [dOG02]

Han et al. presented an interest management approach for NVEs based on so-called *interest groups*. In their scheme, all user avatars are surrounded by a spherical AoI. Users are dynamically grouped in interest groups based on their interests and distance. Update messages are propagated based on these groups. Each user in the group multicasts update messages to the rest of the group whenever it moves or interacts with the world. Each group has one *representative*. If the group is included or overlaps with the AoI of a user that is not part of the group, the representative sends an aggregation of the update messages from within the group with low frequency to this user. Users can not join or create several groups. They have to assign a priority for their interests. Groups are then created or joined based on the priorities. Later on, Han et al. described an extended version of their scheme

that aimed at DVEs. In addition to interest groups, the extended version of their scheme tesselates the virtual environment into regions and sub-regions. A *superpeer* is assigned for each region. Each superpeer manages interest groups and multicast communication within its region. Sub-regions are used to provide users with low fidelity data of objects that are located in the sub-region and can only be seen. [HLL00, HLLH08]

Bharambe et al. proposed *Colyseus*. Colyseus is a distributed architecture for online multiplayer games. In Colyseus, users express their interest in surrounding objects via *range query expressions*. Each user defines one or multiple range query expressions. The AoI of a user consists of the union of its defined range queries. Up-to-date object data is retrieved for all objects within a user's AoI. [BPS06]

In the P2P-MMVE *Solipsis*, which was described by Keller et al., every avatar is surrounded by a 2-dimensional circular AoI, the so-called *awareness area*. The awareness area is used to determine the neighboring avatars for building and maintaining a network topology and to retrieve data about objects that are located within the viewing distance of a user. The radius of the awareness area is variable and depends on the density of users and objects within the awareness area as well as the hardware capabilities of the peer. [KS02, KS03]

Over time, two independent versions of Solipsis were proposed. The described first version of Solipsis aimed at building a network topology with the structure of a torus. Later on, a second version of Solipsis was proposed that uses Delauney triangulation to build a network topology. The second version of Solipsis is described later on in the context of zone-based interest management with dynamic tesselation.

Extended Aura-based Interest Management

Basic aura-based approaches implicitly assume that object state changes or events describing these changes occur at a single point in the virtual environment. A state change is only relevant for a user if the position of the changing object or the other user that triggered the corresponding event is located within the user's AoI. *Extended aura-based interest management* approaches aim at providing a higher flexibility by giving state changes or events their own spatiality. These approaches typically extend the concept of AoI with one or multiple additional spaces that model the extent of an object's state change or an event. This allows a more precise determination of relevancy in comparison to basic aura-based approaches. In extended aura-based

interest management approaches, relevancy is determined based on the intersection of the AoI and the additional spaces.

Morse et al. reported the functionality of *data distribution management in the HLA*. The HLA distributes messages based on multidimensional routing spaces, so-called *update regions* and *subscription regions*. Each space consists of a set of coordinates that can be adjusted dynamically at runtime. Each simulation entity can define multiple update and subscription regions. 'Update regions are associated with individual objects' [MS97, p. 345] while 'a federate might have a subscription region for each sensor system being simulated' [MS97, p. 345]. The data type that will be sent to an update region or received via a subscription region can be specified by a given object class and attribute name or by an interaction class. The relevancy of an object's update is then determined by intersecting the associated update region with available subscription regions of the same data type. [MS97]

Lee at el. proposed *APOLO*, a peer-to-peer overlay network for massively multiplayer online games. In APOLO, each update message has a so-called *coverage area*. A spanning multicast tree is derived from the existing network topology of APOLO based on the coverage area. The update message is propagated to all network nodes in the coverage area. [LLI+05]

Ito et al. presented a peer-to-peer architecture for massively multiplayer online games that propagates information based on virtual spaces. In the described architecture, player avatars are surrounded by so-called *visible areas*. Visible areas move with the avatar in case the avatar changes its position. Virtual environment objects are surrounded by so-called *surveillance areas*. All visible and surveillance areas have the spatial structure of a 2-dimensional square and the same size. Because of the same area size, all players whose avatar is located within the surveillance area of an object are interested in the object. Multicast groups are built based on this information. Object updates are propagated via these multicast groups. [ISST06]

Shun-Yun Hu described a communication mechanism for MMVEs called *spatial publish subscribe*. In a spatial publish subscribe system, each node specifies a so-called *publication space* and a so-called *subscription space*. These spaces are updated according to node movement. Nodes send messages to the publication space. The system checks for intersections with the subscription spaces of other nodes and delivers the messages to all nodes with an intersecting subscription space. Matching of publication and subcription spaces and delivery of messages is performed by so-called *interest matchers*. Interest matchers are assigned to participating nodes based

on given criteria, for example based on a given tesselation of the virtual environment. In this case, one interest matcher is assigned for each zone and performs the matching of spaces for this zone. In case a publication space intersects other zones, the interest matcher of the original zone forwards the messages to the interest matchers of the intersected zones. [Hu09]

Zone-based Interest Management

Aura-based and extended aura-based interest management approaches use spatiality as main criterion for interest management. User interest and the extent of update messages or events are represented by spaces that are either attached to entities of the MMVE, for example user avatars or objects, or to entities of the system, for example update messages or events. *Zone-based interest management* approaches also use spatiality as main criterion for interest management. In contrast to aura-based and extended aura-based approaches, zone-based approaches define the spatial structures that are used for interest management based on the virtual environment. Zone-based approaches typically tesselate the environment into *disjoint zones*. User peers are then assigned to these zones, for example according to their avatars' positions. Update messages are propagated to user peers based on the zones, for example by sending update messages via multicast to all user peers whose avatars are located within a certain zone.

In comparison to aura-based and extended aura-based approaches, zone-based interest management approaches only allow a less precise determination of relevancy because zones are only a rough approximation of an individual user avatar's interest. However, the provision of individual auras for all user avatars and determination of interest based on these auras is a computation-intensive task. This is even more true for extended aura-based approaches because these approaches include additional spatial structures for the influence of state changes or events and demand the calculation of a potentially very large number of intersection operations for determination of relevancy. Using zones is less accurate, but at the same time has the potential to be much less computation-intensive than using individual auras or auras extended by additional spaces.

Some proposals combined aura-based and zone-based interest management. The previously summarized publication by Shun-Yun Hu about spatial publish subscribe [Hu09] adds conceptual elements from zone-based interest management to the extended aura-based approach of spatial publish subscribe by proposing the use of

a tesselation of the virtual environment for the assignment of interest matchers. The P2P-MMVE VON, which is described later on in this work, tesselates the virtual environment into disjoint zones and, in addition, uses individual AoIs that are intersected with the zones to build and maintain a peer-to-peer network topology.

Existing zone-based interest management approaches can be further classified into two sub-categories: 1) Approaches that use a *static tesselation* algorithm. 2) Approaches that use a *dynamic tesselation* algorithm. Static tesselation algorithms define or calculate a partitioning of the virtual environment before runtime. Only the assignment of user peers to zones is performed at runtime. Dynamic tesselation algorithms partition the virtual environment at runtime. Dynamic tesselation algorithms often include mechanisms that adjust the tesselation scheme dynamically for further optimization of update propagation. In the following, this work first gives an overview of approaches with static tesselation algorithms. After that, it presents approaches with dynamic tesselation algorithms.

Macedonia et al. presented an interest management approach for use with *NPSNET*. In their approach, the virtual environment is tesselated into static, fixed-size, hexagonal zones. An application layer multicast group is created and maintained for each zone. Entities, for example user avatars or vehicles, can join several multicast groups depending on the zones that they are interested in. In case an entity moves, the multicast groups are updated according to the entity's interest. [MZP+95]

Knutsson et al. proposed a system architecture that supports massively multiplayer online games with peer-to-peer communication. Their approach includes an interest management scheme that partitions the virtual environment into static, fixed-size zones. Each player is assigned to a zone based on the location of its avatar. A so-called *interest group* is built for each zone. The interest group of a zone is used to multicast messages to all peers within the zone. In case a user leaves a zone and moves into another zone, the interest groups are rearranged. A so-called *coordinator* peer is assigned for each zone. The coordinator is used for update propagation in combination with the concept of interest groups. Position changes of avatars are send directly via multicast to all other players in the zone based on the interest group. Object state changes are first sent to the coordinator. The coordinator resolves potential conflicts. Then it sends the state change via multicast to its zone. [KLXH04]

Iimura et al. proposed a *zoned federation model* for scalable massively multiplayer online games based on peer-to-peer networks. The zoned federation model adds

an additional software layer between the game program and the P2P network, the so-called *zoning layer*. The zoning layer tesselates the virtual environment into static, disjoint zones. Then it selects and assigns peers as *zone owners*. Zone owners perform server-like tasks for their zone, for example the distribution of workload over the participating peers. The authors do not give a certain tesselation algorithm or zone shape. This task is left open for developers which want to use the zoned federation model. [IHK04]

Yu et al. presented MOPAR, a *mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games*. MOPAR tesselates the virtual environment into fixed-size zones of a 2-dimensional hexagonal shape. Peer nodes are divided into *master nodes*, *slave nodes* and *home nodes*. Each zone has one master node, one home node and multiple slave nodes. The avatars of master and slave nodes of a zone are located in the zone. The home nodes are only assigned virtually to zones and do not have to be located in the zone. The home nodes are responsible for selecting master nodes and assigning them to zones. In addition, they establish and maintain connections between the master nodes of neighboring zones. The slave nodes are registered with the master node of the zone where their avatar is located. The propagation of update messages is performed as follows: The master nodes receive all update messages from their zone. In addition, they receive update messages from their neighboring zones via the connections to the master nodes of these zones. Then they provide their slave nodes with updates about the neighborhood of the slave nodes. [YV05]

Yamamoto et al. described a *distributed event delivery method* for MMORPGs. In their method, the game space is tesselated into fixed-size, 2-dimensional squares. Each square is managed by a *responsible node*. User avatars have an AoI that is modeled by a fixed-size, 2-dimensional rectangle. Each user registers with the responsible nodes of all squares that intersect its AoI. State changes are propagated based on events. In case an action takes place, a corresponding event is triggered. The event message first is sent to the responsible node of the square where the action took place. The responsible node then forwards the message to its registered users. The presented interest management approach is further enhanced by aggregation. The responsible nodes collect events and send them as a list to registered users. In addition, so called *load balancing trees* are built within squares in case the number of users within a square exceeds a given threshold. The square is further tesselated into sub-squares. Users are arranged into a tree based on the sub-squares. Events

are propagated to the responsible node via the load balancing tree. [YMYI05]

El Rhalibi et al. described a peer-to-peer architecture for massively multiplayer online games. Their architecture includes an interest management approach that tesselates the virtual environment into fixed-size, rectangular regions. For each region, a peer is assigned the role of a *region server*. A *peer group* is established. The peer group includes all peers whose avatars are located in the region. The peer groups are maintained by the region servers. Update propagation between peers of a region is performed via the peer group. In addition, the region servers are connected with each other. This allows the propagation of updates between regions via the region servers. [ERM05]

Hampel et al. presented a peer-to-peer architecture for massively multiplayer online games. Their architecture includes an interest management approach that tesselates the virtual environment into fixed-size regions. The authors describe an example using 2-dimensional hexagonal regions. Each region has a so-called *region controller*. The region controller acts for this region as a server. A user is interested into a region if its avatar is located in the region. It declares its interest by subscribing to the controller of the region. State changes are propagated based on events. In case an event is triggered because of a game action within a region, it is first sent to the region controller. The region controller then calculates the new game state based on the event. After that, the controller sends out the new game state to all peers that are subscribed for the region. [HBH06]

Ahmed et al. proposed the NVE system *MM-VISA*. MM-VISA tesselates the virtual environment into hexagonal regions of a fixed size. Each region is assigned a so-called *coordinator* peer that manages communication for the region. Users declare interest in a region if their avatar is located in the region. In addition, they declare interest in the surrounding regions up to a given distance value. The coordinators form a top level mesh hierarchy that is used for a hand-over of users between regions and for management of interest of users from other regions. Communication within regions is performed via application layer multicast. Multicast trees are built based on movement characteristics as criteria. Entities that move fast have a tendency to change regions often. Therefore, such entities are located lower in the tree to reduce the needed effort for tree reconstruction in case the entity leaves the region and enters another region. [ASdO06]

Chan et al. presented *Hydra*, a peer-to-peer architecture for massively multiplayer online games. Hydra assumes that the virtual environment is partitioned into dis-

joint regions of a fixed size. The authors do not give an algorithm for partitioning nor mention a certain region shape. They assume that the partitioning is predetermined by the game developer which uses Hydra. Peers participate in the system as clients and, in addition, they are able to perform server tasks for one or multiple regions. Clients can only interact with other clients that are connected to the same server. The authors also delegate load balancing. It is up to the game developer to find an appropriate world partitioning to avoid overloading of servers. [CYB+07]

The *Badumna Network Suite* for massively multiplayer online games by Kulkarni et al. includes an interest management approach that tesselates the virtual space into fixed-size regions. The regions are inserted into a distributed hash table. The peer closest to a region takes over responsibility for the region and performs server-like tasks including propagation of update messages. Depending on the nature of the massively multiplayer online game application, size of the game space and region density, Badumna is able to switch to alternative interest management protocols: The *dynamic bounded protocol* and the *gossip protocol*. The dynamic bounded protocol reduces traffic on the distributed hash table by aggregating and performing the requests of entities within a certain bounding space at once. The gossip protocol is used in case of a very high density of entities. In this case, the entities start communicating directly with its neighboring entities and only query the peer responsible for the region periodically at a reduced frequency. [Kul09]

Hu et al. proposed a *scalable peer-to-peer networked virtual environment*. In their system, the virtual environment is tesselated dynamically into disjoint zones at runtime by using *Voronoi diagrams*. A Voronoi diagram is an existing mathematical construct. 'Given n points on a plane (each point called a site), a Voronoi diagram is constructed by partitioning the plane into n non-overlapping regions that contain exactly one site in each region. A region contains all the points closest to the region's site than to any other site.' [HL04, p. 131]. In the approach by Hu et al., on each peer a 2-dimensional Voronoi diagram is calculated for the virtual environment so that the avatar of each peer node is located as site at the center of a region. Neighboring peer nodes are identified based on the Voronoi diagram and connections between them are established. In case of avatar movement, the Voronoi diagram is recalculated and the connections between neighboring peers are adjusted. In addition to the Voronoi diagram, each avatar has a 2-dimensional circular AoI. Interest is determined by combining the regions of the Voronoi diagram and the AoI. All neighboring peers that are located inside the AoI are interested in updates

from the corresponding user avatar and have to be provided with update messages. [HL04]

Later on, Hu et al. described a so-called *Voronoi-based overlay network* (VON). VON includes an extended version of the approach described in [HL04]. The original approach with the same fixed-size AoI for all peers showed a potential weakness when confronted with a high user density at certain places because this leads to a lot of connections and update messages for the peers. The extended approach includes dynamic-sized AoIs. An individual AoI radius is calculated for each peer based on the peer's networking capabilities. The AoI is adjusted dynamically. If the number of connected neighbors exceeds a given threshold, the AoI is shrinked. The AoI is restored if the number falls below the threshold. [HCC06]

Analogical to VON, the second version of Solipsis by Frey et al. tesselates the virtual environment by using Voronoi diagrams. *Solipsis II* builds on *RayNet* [BKR07]. RayNet is a peer-to-peer overlay network that tesselates the environment based on Voronoi diagrams and creates and maintains a network topology according to a *Delauney graph*. A Delauney graph connects all sites of a Voronoi diagram. As a result, the peer nodes are connected to their neighbors and are able to communicate with them. The network topology is adjusted according to avatar movement. Solipsis II further includes an AoI concept. All avatars are surrounded by so-called *bounding boxes*. Bounding boxes are defined as spheres. The exchanged informations are categorized into different levels of detail. Only the peers of avatars with overlapping bounding boxes exchange informations in high-grained detail. The other neighboring peers receive a low-detailed version. [FRP+08]

Douglas et al. presented a peer-to-peer architecture for massively multiplayer online games. In their architecture, the 2-dimensional virtual environment is tesselated into squares. A *distributed spatial data service* is established over a peer-to-peer network based on the tesselation. The data service is arranged as a *quad tree*. In case a square gets overloaded, it is dynamically subdivided into four smaller squares and the quadtree is updated. The data service is used to discover and query relevant entities for a certain peer. In addition to the data service, the approach includes an *entity interaction service*. Every entity registers a so-called *event region* with the system. The event region is defined so that it guarantees to contain the entity for a period of time. The size of the event regions for peers can vary. An entity can move a certain distance within its event region while the center of the region stays fixed. If the entity exceeds a threshold distance from the center, the event region is updated.

All event regions are stored in the quad tree for discovery and query. Entities are likely to interact with each other if their event regions overlap. Therefore, direct connections are established between two entities if intersections are detected. After that, updates are exchanged directly. [DTHK05]

GauthierDickey et al. proposed to manage interest based on *n-trees*. N-trees are defined as 'a generalization of the octree ... that recursively subdivide an n-dimensional space' [GLZ05, p. 87]. In their approach, the virtual environment is tesselated into n sub-domains based on n-trees. These sub-domains can be further recursively subdivided at runtime in case there is a high density of entities within a sub-domain. GauthierDickey et al. described a game that includes a 2-dimensional virtual world. The corresponding n-tree is used to organize peers by their application-level scope of interest and to propagate events between peers. Each peer has a function that describes its *scope of interest* and joins the tree based on the result of the function. Each event has a function that describes its *scope of impact* and influences one or multiple sub-domains of the tree. In case a peer generates an event, it uses the tree to propagate the event to other peers. The event's scope is intersected with the sub-domains to determine the influenced sub-domains. Then the event is propagated to all peers that are connected to the corresponding nodes of the influenced sub-domains. [GLZ05]

Please note that the propagation algorithm of the interest management approach by GauthierDickey et al. shows similar characteristics as the approaches that were presented under the category of extended aura-based interest management. The function for the scope of interest can be interpreted as AoI and the function for the scope of impact can be interpreted as a spatial representation of the event's influence. However, the approach by GauthierDickey et al. is classified as zone-based approach with dynamic tesselation because the matching between the functions is performed based on the n-tree structure as main conceptual element.

Morillo et al. described a DVE system called *COVER*. In COVER, the virtual environment is tesselated into square regions. A peer is assigned as *supernode* for each region. Supernodes are arranged by a *dynamic quadtree structure*. In case the resources of a supernode are used to full capacity, the region is further subdivided into squares and each subregion is assigned a new supernode. The supernodes of the subregions are then added to the quadtree structure. COVER additionally uses the AoI concept to determine neighboring avatars and to make sure that these avatars are aware of each other's state. Every avatar has a 2-dimensional circular

AoI. The AoI is used to determine different types of neighbors. An avatar is a so-called *first level neighbor* of another avatar if the avatar is located within its AoI. All first level neighbors of an avatar's first level neighbor are so-called *second level neighbors* of the avatar. Each time an avatar moves, it sends a message to its first level neighbors. The first level neighbors then propagate the message to the second level neighbors. All avatars are classified into *covered* and *uncovered* avatars. An avatar is covered if its first and second level neighbors are located in such a way that the intersections of their AoIs totally cover the AoI of the avatar. Otherwise, the avatar is uncovered. For covered avatars, awareness is provided automatically because no avatar can approach them without being detected by the neighbors. For uncovered avatars, awareness is provided by using the supernode structure. In case of movement, uncovered avatars send a message to their neighbors and, in addition, to the supernode of the region. [MMOnD06]

Interest Management based on Visibility

Interest management approaches based on auras or zones represent the spatiality of user interest or the influence of events with basic mathematical structures, for example circles, rectangles or hexagons. This works well in outdoor scenarios because basic mathematical structures are able to approximate the spatiality of user interest and the influence of events in an accurate way. However, these approaches can lead to shortcomings when confronted with *indoor scenarios*, for example user interactions within buildings. In indoor scenarios, the avatar view is often limited by obstacles such as walls. An aura or a large zone does not necessarily approximate the spatial structure of a room in an accurate way because it can exceed the size of the room. As a result, the corresponding peer is provided with update messages from other rooms within its AoI or zone, even its avatar is not able to view the actions in these rooms because its view is limited by walls. Interest management approaches that consider the specific characteristics of indoor scenarios have the potential to perform better than aura-based or zone-based approaches in such scenarios.

The common idea of these approaches is to tailor the interest management scheme to the spatial characteristics of the virtual environment. Instead of defining spatial structures for interest or influence and using these structures at runtime to determine relevancy, these approaches analyze the virtual environment beforehand and calculate *visibility relations* between existing parts of the virtual environment. Assuming an indoor scenario within rooms, visibility relations are calculated and stored that

define the potential visibility of other rooms from certain spots in the building. At runtime, the interest of a user avatar is determined based on the stored relations. Instead of calculating intersections between spatial structures, the user computers perform a database query on the stored relations. The corresponding peer is then provided with update messages from the visible rooms. Some approaches use a negation of this idea and calculate *non-visibility relations*. Instead of making a query on the visibility relations to determine the interest in rooms and then providing a peer with update messages from these rooms, these approaches use the non-visibility relations to filter out update messages from rooms that can not be seen. In the following, this work gives an overview of interest management approaches based on visibility and non-visibility relations.

The *potentially visible sets* (PVS) approach, which was described by Airey et al. and by Teller et al., analyzes the virtual environment and determines cells, for example according to the rooms of a building. Visibility relations between cells are determined based on the characteristics of the building, for example based on openings or portals between the rooms. The PVS approach computes for each cell the other cells that are visible from the cell and stores the potentially visible cells in a set. At runtime, the potentially visible set of a cell is used to determine the cells that have to be rendered for avatars that are located in the cell. [ARB90, TS91]

Makbily et al. proposed an approach based on *update-free regions* (UFR). The UFR approach can be interpreted as a negation of the PVS approach. Mutually irrelevant UFRs between pairs of users are calculated based on the criteria proximity, visibility and direction. The UFR approach varies from the other described approaches based on visibility because the calculation of UFRs between user pairs is performed at runtime and UFRs are recalculated in case of avatar movement. After UFRs were calculated, update propagation between a pair of users is optimized based on the result. As long as the avatars of both users remain in the UFRs, no communication between them is required. If an avatar leaves its UFR, updates are sent and UFRs are recalculated. [MGBY99]

The *frontier sets* approach by Steed et al. can also be interpreted as a negation of the PVS approach. The frontier sets approach uses a PVS structure to generate a new structure of relations that allows a pair of entities to ensure that no interaction between the entities is necessary. In the approach by Steed et al., a pair of cells has a frontier if the cells are mutually invisible for each other. A frontier set includes all frontiers of a cell. Update messages are filtered based on the frontier set. A user

which is located in a cell does not receive updates from another cell if both cells have a frontier. [SA04]

The early version of the frontier sets approach, which was described in [SA04], included pre-computation and local storage of frontier sets on all peers. Later on, the authors realized that this requires a massive amount of storage space, at worst up to n^3. Therefore, they extended their approach with an *enhanced PVS structure*. The enhanced PVS structure is pre-computed, stored and used as an intermediate data structure. It needs less storage space, at worst up to n^2. At runtime, frontiers are calculated based on the enhanced PVS structure when necessary. [SA05, SZ08]

Frontier sets show strong similarities to UFRs. In their later work, the authors even compared their approach to UFR as a 'concrete example of a more general class of algorithm called update-free regions' [SZ08, p. 25].

Interest Management based on Content Attributes

Aura-based, zone-based and visibility-based interest management approaches all put an emphasis on the aspect of spatiality when deciding about the relevance of update messages or events. However, it is possible that users are only interested in updates about certain attributes of objects in their surroundings. In the context of interest management, this notion can be used for a fine-grained filtering of update messages based on *content attributes*. In the following, this work describes two systems that include such interest management approaches.

Bharambe et al. presented *Mercury*, a distributed publish subscribe system for internet games. Mercury includes a subscription language that allows users to perform subscriptions into certain attribute types. Subscriptions are a *conjunction of predicates* that are described by tuples of the form (type, attribute, operator, value). In case of state changes, publications are made. Publications include a list of *typed attribute-value pairs* that are described by tuples of the form (type, attribute, value). Relevancy is determined by matching publications with existing subscriptions. In case one of the attribute-value pairs of a publication falls within the value range defined by one of the predicate tuples of a subscription, the corresponding user is provided with the publication. Matching of publications and subscriptions and routing of messages are performed in a distributed way by so-called *attribute hubs*. Each of these hubs is responsible for the matching of publications and subscriptions of a certain attribute. [BRS02]

The P2P-MMVE system *Donnybrook* by Bharambe et al. focuses on fast-paced first person shooting games. The included interest management approach makes use of the limits of human cognition. It builds on the assumption that in first person shooting games users typically focus attention on only a small number of other user avatars. The authors argue that only these avatars have to be displayed with a tight consistent state, while the rest of the avatars within a user's field of view can be displayed less consistent. Therefore, Donnybrook calculates so-called *interest sets* for every user at each frame. The interest set of a user contains the five users in which the user has the highest interest. The grade of interest is determined based on three weighted criteria: 1) User proximity, because users which are close by each other are highly interested into each other. 2) Aim, because users have an orientation and tend to be interested highly into users at which they are aiming. 3) Interaction recency, because users which interacted recently are highly interested into each other. A user receives updates from other users within it's interest set once every frame. The states of avatars of users which are not included in the user's interest set are not updated regularly. In case the avatars of other users are not included in the interest set of a user, but are located within the field of view of this user, these avatars are represented by bots. The bots act based on given artificial intelligence routines and emulate user behavior. [BDL+08]

2.2.2. Dead Reckoning

In case a user avatar or an object of the MMVE moves on one of the participating user computers, position updates have to be sent to other user computers that hold a copy of the avatar or object in order to adjust the state of the copy. In a MMVE based on a client/server architecture, the server is responsible for distributing these updates. In a P2P-MMVE, these updates are sent directly to other user computers.

Dead reckoning approaches aim at reducing the network traffic for propagation of position changes to other user computers based on the following notion: Instead of sending the position updates of moving objects regularly, only the *initial position* and *additional informations describing the movement*, for example the direction and speed of movement, are sent to user computers with a copy. After that, future position changes are extrapolated based on mathematical equations and the additional informations and are applied to the copy.

Several approaches optimize dead reckoning by using a *deviation threshold*. The server or the peer that holds the original object also performs an extrapolation of

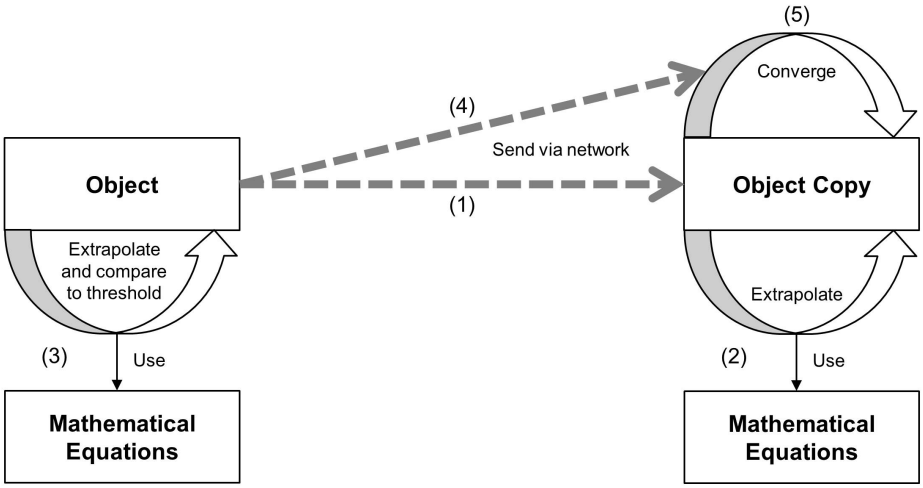


Figure 2.2.: Basic concept of dead reckoning

the future positions based on the same equations and informations as the computers with a copy. Based on the result, it checks if the current position of the original object deviates from the extrapolated position and compares the deviation with a given threshold. In case the threshold is exceeded, the current position of the original object and an up-to-date version of the additional informations is sent to all clients or peers with an object copy.

In order to avoid erratic position changes after the current position and up-to-date version of the additional informations were received, dead reckoning approaches often do not apply the current position of the original immediately to the locally stored copy. Instead, they *converge* the position of the copy slowly towards the received position of the original object for a more realistic user experience.

Figure 2.2 illustrates the basic concept of dead reckoning independently from a specific approach. On the left, the figure displays the original object that is held by the server or a peer. On the right, the figure displays one of the object copies that are held by clients or other peers. In case the original object starts to move, the initial position and additional informations describing the movement, for example direction and speed, are sent to the clients or peers with an object copy (1). After that, the clients or peers extrapolate the future positions of the object copies based on mathematical equations and the sent informations (2). The holder of the original object also performs the extrapolation (3). It checks regularly if a given deviation threshold between the position of the original object and the results of the extrap-

olation is exceeded. In case the threshold is exceeded, the current position of the original object and an up-to-date version of the additional movement informations are sent to all holders of object copies (4). After receiving the position and informations, the holders of object copies converge the state of their local copy of the object based on given algorithms (5).

Dead reckoning is a state-of-the-art concept that is used today by most MMVEs or networked computer games [SKH02]. Dead reckoning has the potential to reduce network traffic because position update messages can be held back while displaying dead reckoned versions of moving objects on the user computers. As a side effect, dead reckoning can also be used to smooth the display of movement in case of high network latency. While waiting for the next position update, the avatar or object copy can be dead reckoned and moved accordingly [Aro]. Over time, numerous dead reckoning approaches as well as optimizations and extensions to the previously described basic concept were proposed. In the following, this work gives an overview of selected work about dead reckoning.

A very early approach that uses conceptual elements of dead reckoning was proposed in the context of distributed multiplayer online games. In 1985, Berglund et al. proposed *Amaze*. In Amaze, users move their avatars that look like monsters through a maze with the goal to shoot the avatars of other users with rockets. In the proposed system, each user computer has to display the maze, the position and state of each avatar as well as the position and state of each fired rocket. In order to enhance the scalability of the distribution of game states between user computers, Amaze includes a technique that the authors call *state extrapolation with correction*. Instead of sending state updates regularly, Amaze only sends a so-called *first derivative* of the game state. It communicates the current and next directions, the velocity, the corridor the object is traveling and the intersection ahead for avatars and rocket objects to other user computers. Based on these informations, future states are extrapolated without any further sending of state updates. A new update is only send in case of a new keyboard input by a user. To keep inconsistency between user computers as small as possible, Amaze further includes a correction mechanism. The extrapolated states are corrected periodically in a fixed interval to avoid strong deviations between the user computers. [BC85]

In the *SIMNET* project for distributed military simulations, dead reckoning is based on *contracts* among the participating simulation nodes. The nodes agree about a certain *deviation threshold* between the state of local objects and extrapolated states

of object copies on other nodes as well as a *retransmission interval*. In case the threshold is exceeded, the nodes guarantee a resending of state updates within the interval. In SIMNET, state updates include the corrected position, velocity and heading information. SIMNET further includes a smoothing technique. In case a state update is lost completely, the state extrapolation for an object copy continues for a few additional seconds. [MT95]

The *IEEE Standard for Distributed Interactive Simulation - Application Protocols* (IEEE Std 1278.1-1995) describes the dead reckoning approach that is included in DIS. Because DIS builds on SIMNET concepts, the described dead reckoning approach shows strong similarities to the approach of SIMNET. In addition, DIS includes *rotational dead reckoning*, a technique that predicts an object's 3-dimensional orientation. In DIS, each participating simulation node holds the original of its own avatar and a local copy for dead reckoning purposes. All nodes hold copies of other avatars that are of interest for them. Initially, a state update about an avatar is sent to all nodes with copies and thresholds for position and orientation are established. In the following, the node with the original and all nodes with copies extrapolate the position and orientation. In case the position or orientation of the original avatar deviates more than the established threshold from the local copy for dead reckoning purposes, the node with the original sends a new update message to the nodes with copies and the copies are updated. DIS uses convergence algorithms in order to minimize erratic position changes in case a position or orientation is updated. The DIS standard allows the use of varying types of dead reckoning formulas. It includes a notation for dead reckoning and describes dead reckoning mathematics. [iee95a]

Over the years, several optimizations and extensions for the dead reckoning approach of DIS were proposed. For example, Lee et al. described an *adaptive dead reckoning algorithm for DIS systems*. Their algorithm aims at increasing the network traffic reduction of dead reckoning by adapting the error threshold dynamically based on spatial areas and by selecting an appropriate extrapolation equation for dead reckoning based on movement characteristics. The threshold is adapted based on the concept of AoI from interest management. The concept of AoI is extended by a so-called *sensitive region*. A sensitive region has a smaller radius than the AoI. In case an avatar steps inside the sensitive region of another avatar, it is very likely that a collision will occur. Depending on whether the AoIs of avatars 1) do not overlap, 2) do overlap, 3) an avatar moves into the other avatar's AoI or 4) an avatar moves into the other avatar's sensitive region, an appropriate threshold is selected. For

the selection of an appropriate extrapolation equation, the movement of an avatar is classified into the types smooth, bounce and jolt. The extrapolation equation is chosen according to the type. [LCTC00]

Chen et al. presented another optimization proposal for dead reckoning in DIS. Their *fuzzy dead reckoning algorithm for Distributed Interactive Simulations* adapts the error threshold dynamically at runtime based on fuzzy logic. Chen et al. argue that in addition to distance between avatars other types of relations should be considered as well for adapting the error threshold. Based on the relationships between avatar properties and a priority for each property, a so-called *fuzzy correlation degree* is calculated and the error threshold for dead reckoning is adjusted accordingly. [CC05]

In contrast to DIS, the run time infrastructure of the HLA does not support dead reckoning directly. Moreover, dead reckoning techniques have to be implemented by the federates [Fuj98]. Lin et al. pointed out that the HLA 'no longer dictates the set of formulas that can be used' [LBW97, p. 104] and 'the concept of dead reckoning is extended to attribute extrapolation' [LBW97, p. 104]. They presented a dead reckoning translator between DIS-compatible simulators and the HLA run time infrastructure. Dead reckoning informations are extracted from DIS protocol data units and sent via the HLA run time infrastructure. After receiving dead reckoning informations via the HLA run time infrastructure, the translator converts the informations back into protocol data units for DIS-compatible simulators. [LBW97]

NPSNET-IV includes an approach for dead reckoning of player positions based on the 'players and ghosts paradigm' [MBZ+95, p. 94]. The avatar of each player is represented on other computers by so-called *ghosts*. After an initial update message including location, orientation and velocity was sent over the network, the ghosts update their positions based on dead reckoning without further sending of network messages. The computer that hosts the original avatar also performs the dead reckoning extrapolations and compares the actual position with the predicted positions of the ghosts. A new network message with an up-to-date location, orientation and velocity is sent either after an error threshold is exceeded or 5 seconds have passed. [MBZ+95]

Duncan et al. presented a *pre-reckoning algorithm for DVEs*. They aimed at optimizing the accuracy of existing dead reckoning algorithms for DVEs. According to the angle of turns during movement of the original avatar, their approach anticipates possible future exceedances of the error threshold on the computer that holds the original dead reckoned avatar. In case a potential exceedance in the near future is

detected, an up-to-date state is sent over the network to all computers that hold copies of the avatar before the threshold is actually exceeded. The approach by Duncan et al. potentially results in a higher accuracy at the expense of an increased network traffic. [DG03]

The *Badumna Network Suite* for massively multiplayer online games has a built-in dead reckoning approach that extrapolates the current position of an entity based on direction and velocity. It is up to the application developers which use Badumna to decide when a new up-to-date position should be sent. This has to be performed explicitly by the massively multiplayer online game application. The provided dead reckoning approach works under the assumption of a fixed velocity. [Sca]

Yahyavi et al. proposed an approach called *AntReckoning*. They aimed at increasing the accuracy of dead reckoning concepts that are based on mathematical formulas. The approach by Yahyavi et al. considers user interest as an additional criterion for prediction of future avatar positions. All entities in the virtual environment, such as other users, objects or locations, are assigned a value that describes their attractiveness. Over time, the entities spread and fade their attraction within the virtual environment similar to the spreading of pheromones within ant colonies of the real world. The authors presented a formula that extends the typical mathematical formulas for dead reckoning by incorporating the attractiveness of nearby objects via so-called *attraction forces*. [YHK11]

Singhal et al. described a *position history-based protocol for distributed object visualization* that extends basic dead reckoning approaches by taking previous position changes of objects into account for dead reckoning. Their protocol observes the movement of an object over time and then selects an appropriate formula for dead reckoning based on previous position changes. The up-to-date position of the original object is resent either after an error threshold is exceeded or a given period of time has passed. [SC94]

Later on, Singhal et al. described an approach for scalability enhancement in distributed interactive simulations that combines dead reckoning with aggregation. The approach aggregates objects of the same organization type that are located within the same zone of a virtual environment into groups. These so-called *projection aggregations* are assigned a common multicast address. Users which are located far away from the projection aggregation or which have only low interest into the projection aggregation can subscribe via the multicast address. Subscribed users are provided with a rough summary of all contained objects from the projection aggregation every

few seconds instead of fine-grained updates. Each summary contains the number of objects, a summary position and information about how the objects are distributed in dependency of the given location. The summary position is calculated as average of the positions of the objects that are contained in the aggregation. The object distribution is described by the radius of a bounding sphere that contains all object positions as well as the mean and standard deviation of the object positions from the summary position. On the computers of the users which receive a summary, the objects of the projection aggregation are then placed randomly within the defined sphere based on the deviation values. Over time, the target computers perform dead reckoning for the summary point, radius and mean distance and adjust the object positions accordingly after a new summary was sent. Projection aggregations are created dynamically and are implemented as logical entities within the system. User computers can detect existing projection aggregations within a region by submitting a query to the system. [SC96]

Capin et al. described a *dead reckoning technique for streaming virtual human animation*. Their approach performs dead reckoning for the joint angles of body parts of avatars. Instead of sending fine-grained information about the movement of certain body parts of the original avatar, for example the movement of an arm, dead reckoning formulas are applied to the angles and the avatar copies move their body parts accordingly. [CET99]

2.2.3. Prediction Techniques for Avatar Behavior

In addition to movement, user avatars often perform other actions in the virtual environment. The copies of an avatar on other user computers have to act accordingly. Similar to movement, updates have to be sent over the network to propagate the avatar actions.

Over time, several approaches were proposed that aim at reducing the network traffic for sending the outcome of avatar actions. These approaches use a similar notion as dead reckoning. Instead of sending multiple precise update messages for particular avatar actions over the network, only a rough descriptive information is sent and the avatar copies act autonomously based on *prediction techniques for avatar behavior*.

Figure 2.3 on page 36 gives an overview of the basic concept. On the left, the figure displays the original avatar. On the right, the figure displays one of the avatar copies. Instead of sending numerous fine-grained updates describing avatar actions,

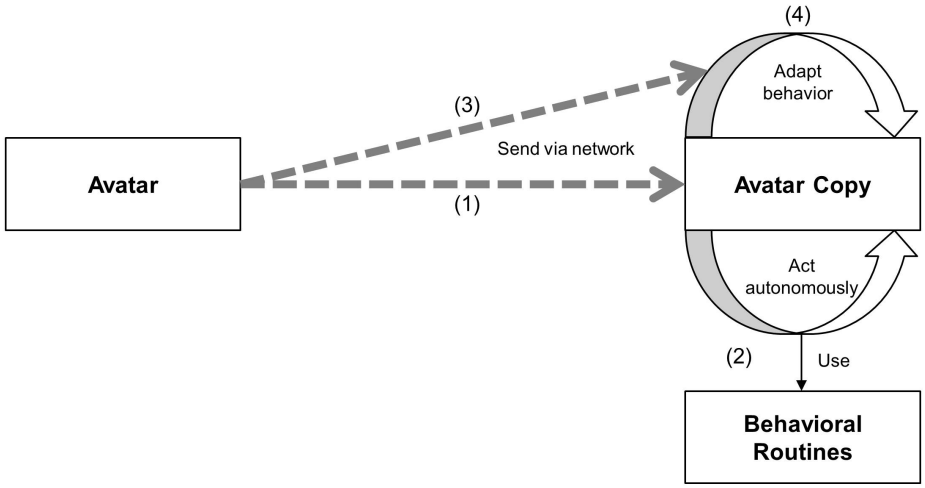


Figure 2.3.: Basic concept of prediction techniques for avatar behavior

only a rough description of the goal, outcome or behavior is sent to other computers with avatar copies (1). After that, the avatar copies act autonomously based on given behavioral routines and the received description (2). Some approaches further try to optimize this concept and minimize deviations by resending up-to-date states or descriptions in given time intervals (3) and by adapting the behavior of the avatar copies accordingly (4). In the following, this work gives an overview of approaches that reduce network traffic based on the described concept.

Shi et al. proposed the concept of *smart avatars*. The original avatar of the local computer is represented on other computers by smart avatars. Instead of sending specific state changes as multiple updates over the network, for example to describe specific movements of body parts for an action, the computer that holds the local avatar sends text messages including a rough description of the executed actions. One of the examples given by the authors is 'take the ball' [SSGB99, p. 159]. Instead of representing each specific movement for taking the ball by an update and sending the updates over the network, only the description is sent. After receiving the description, the avatar copies take the ball autonomously. [SSGB99]

Szwarcman et al. presented a framework for *networked reactive characters*. Their framework includes an approach that aims at reducing network traffic for propagation of avatar actions based on the idea of *autonomous clones*. The avatar on the local computer is called *pilot*. The pilot defines a general goal for the following avatar behavior, for example 'walk to the wall between the doors' [SFC00, p. 205].

The pilot sends this information to its copies on other computers. The copies are called *clones*. The clones then act autonomously towards the given goal. In order to reduce the deviation between the copies and the pilot, the approach includes a recovering mechanism. Pilots resend state updates in intervals. In case a clone detects a strong deviation, it adapts to the current state of the pilot. [SFC00]

The P2P-MMVE system Donnybrook by Bharambe et al. emulates the behavior of an avatar on other peers based on artificial intelligence routines. As described earlier in the subsection about interest management, Donnybrook calculates interest sets for each user. The interest sets contain the five users in which the user has the highest interest. To reduce network traffic, this concept is extended by a prediction technique. Avatars of users which are not included in the interest set of a user but are located within the field of view of this user are represented by bots, so-called *doppelgängers*. The doppelgängers receive guidance information from the original user avatar once per second. In between, the doppelgängers use the guidance information to act based on given artificial intelligence routines. [BDL+08]

The authors called this prediction technique based on doppelgängers and artificial intelligence routines *guided artificial intelligence*. A detailed description and definition of guided artificial intelligence can be found in [PUL07]. In the context of Donnybrook, guidance is defined as 'a compact summary of his (a player) predicted behavior for the period between now and the next anticipated message, such as where he expects to go, whom he is targeting, and how often he fires his weapon' [BDL+08, p. 392].

2.2.4. Geocast

Geocast is a concept from mobile and context-aware computing that is used to send messages to certain geographical areas in mobile networks [DR03]. Similar to interest management in MMVEs, the basic notion of geocast is that specific types of messages, for example warning messages, are only interesting for users of mobile devices within certain areas. The messages are propagated to these areas and are delivered to mobile devices based on their location as criterion of interest.

For delivery of messages, geocast has to rely on an underlying location model. For example, Dürr et al. proposed a *hybrid location model* that combines a symbolic location model and a geometric location model. The symbolic model provides addressing of messages based on a hierarchy of symbolic locations, for example to

address a certain room within a certain floor within a certain building. The geometric location model provides addressing of messages based on 2,5-dimensonal spatial areas. A 2.5-dimensional spatial area consists of a 2-dimensional figure as base, a fixed height and an altitude value for the base. A symbolic location can be translated into a geometric location and vice versa. [DR03]

Although not directly related to the presented approach, the concept of geocast might be an interesting starting point for future research. The concept of geocast shows strong similarities to spatial publish subscribe communication in MMVEs. Both, geocast and spatial publish subscribe, address messages via spatial areas. Location models of mobile computing (see [BD05] for a comprehensive overview) might be applicable in the context of a virtual environment or for the implementation of a spatial publish subscribe system. Because spatial publish subscribe is the target communication model of this work, the continuous events approach might be adapted for use case scenarios from the fields of mobile and context-aware computing with geocast. The support of such use case scenarios is beyond the scope of this work. Nevertheless, the concept of geocast is relevant for future work. Section 6.4 of this work discusses potential directions of future research. This includes potential research in the context of geocast.

2.2.5. Own Related Work

An early version of the continuous events approach was presented by Heger et al. at the IEEE Consumer Communications and Networking Conference 2012 [HSS+12]. The work that was presented at the conference described the basic idea of continuous events, defined a formal model for continuous events and gave early versions of algorithms for continuous event execution and management. Compared to this work, no peer crashes, disconnections or overloaded peers were considered. In addition, the work that was presented at the conference was restricted to a single zone, did not include infinite continuous events and no mechanism for modifying existing continuous events in the system after their creation. The simulations had a preliminary character. They were performed to give a first hint at the potential of the continuous events approach and to explore potential parameters for more sophisticated evaluations such as the evaluation included in this work.

2.3. Target System Model

This work originated from the project *Peers@Play* [PaP] that aimed at providing a framework for MMVE developers based on the peer-to-peer communication model. Being part of a project implies the existence of an overall system model that has to be considered in the context of this work. In addition, several MMVE and networking services exist as part of the framework that are used by this work. This section gives an overview of the system model, as considered by this work, and existing services.

Peer-to-Peer Communication Model

The MMVE system consists of a large number of end-user computers (*peers*). These peers are connected via a *wide area network*, preferably the Internet. Every peer can enter and leave the system *intended* at any time. In addition, a peer can leave the system *unintended*, for example due to hardware failures or network outages.

The MMVE system is based on the peer-to-peer model. The network communication between the peers as well as the provision of the virtual environment and system tasks are *operated cooperatively* by the peers. Therefore, each peer is potentially able to take over tasks for *communication* (such as propagation of state updates to other peers) and *operation* (such as persistent storage of object states or calculation of state updates according to user input).

Tesselation into Zones and Assignment of Superpeers

The target P2P-MMVE system makes use of so-called *superpeers*. Peers with strong hardware and network capabilities are selected from the set of all available peers and are assigned additional system tasks for running the MMVE. The assignment of tasks to superpeers is done spatially, based on a *tesselation* of the virtual environment. The virtual environment is tesselated into disjoint zones and for each zone a superpeer is selected and assigned to this zone. A superpeer is responsible for providing system tasks for its zone.

The target system uses a 2-dimensional projection of the virtual environment and tesselates the projection into disjoint zones based on a given tesselation algorithm.

Stateful Objects

The virtual environment consists of a large number of *stateful objects*. Each object has a specific *location* within the virtual environment. In order to display the virtual environment, a copy of all objects in the avatars' surroundings has to be provided on each peer. The object state can change, for example because of an interaction of an avatar with the object. In order to have an up-to-date state of an object on all peers, state changes have to be *propagated* to other peers via network messages and applied to the object copies.

Event-Driven Propagation of State Changes

The propagation of state changes of the target system is *event-driven* and, therefore, decoupled from the objects. In case a user input is detected, the input is translated into an avatar action. Then the action is performed by the avatar and a corresponding event is generated by the system. The event is propagated via the network to other peers based on *relevancy* to make sure that all peers for which the event is relevant are provided with the information contained in the event. In case a peer receives an event from another peer, the state changes described by the event are applied to the locally stored object copies.

Event Propagation based on Spatial Influence in the Virtual Environment

The relevancy of an event is determined based on the *spatial influence* of the corresponding user action and the *spatial interest* of users in a 2-dimensional projection of the virtual environment. The influence of a user action is described by a so-called *area of effect* (AoE) that consists of one or multiple 2-dimensional spatial shapes in the virtual environment. Every user avatar is surrounded by another spatial shape, the so-called *area of interest* (AoI). An event is relevant for a peer if the AoI of the avatar of that peer intersects at least one of the shapes of the event's AoE (see [SSSB09] and [HSSB09]).

Figure 2.4 on page 41 illustrates the determination of relevancy based on AoEs and AoIs. The user avatars A, B, C, D and E each are surrounded by a circular AoI. The AoI size of avatars can vary, for example avatar A has a larger AoI than avatar E. In case an action is performed, an event representing the action is created, the

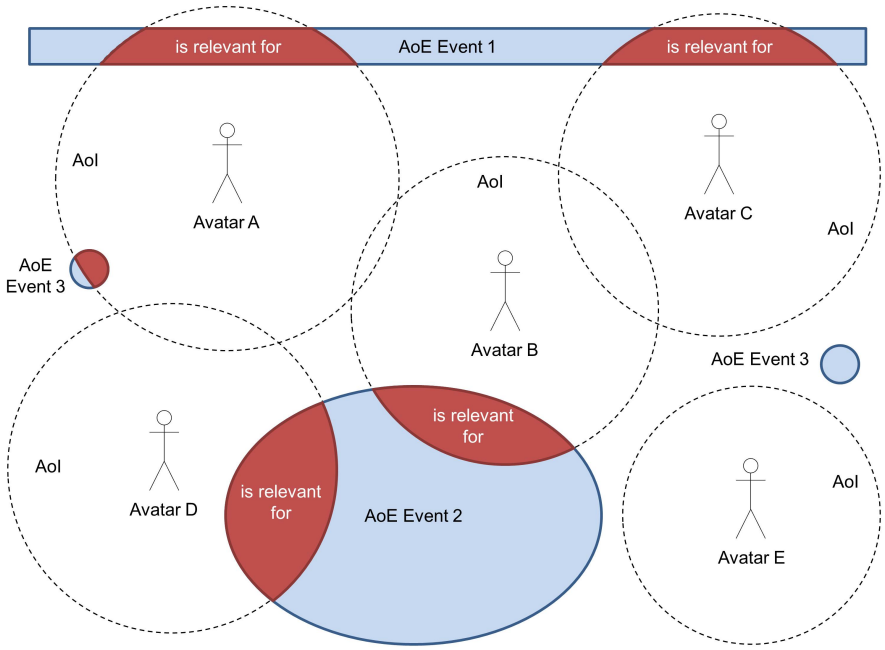


Figure 2.4.: Propagation of events based on areas of interest and effect

spatial influence of the action is determined and the event is propagated to an AoE corresponding to the spatial influence. The figure shows the AoEs of events 1, 2 and 3. The AoE of an event can consist of several spatial shapes, for example event 3 represents a teleport of an object from one location to another location in the virtual environment and, therefore, the AoE consists of two areas with the shape of a small circle at the original spot and at the target spot. An event is relevant for a user and has to be propagated via the network to this user's peer if the AoI of the user avatar intersects at least one of the shapes of the event's AoE. For example, the AoE of event 1 intersects the AoIs of avatars A and C and, therefore, the event has to be delivered to the peers that correspond to these avatars.

The superpeers have an important role for the determination of relevancy and propagation of events. Regular peers are only provided with spatial subscriptions from other peers up to a certain range, the *area of propagation* (see [SSSB09]). Therefore, a single peer is only able to determine relevancy of events for other peers up to a certain distance. In order to make sure that events can be propagated to AoEs outside of a peer's area of propagation, the superpeers are provided with all spatial subscriptions for their zones and are used as a fall-back solution for determination of relevancy. If a peer determines that it is not able to propagate the event directly, it uses the superpeer for propagation. Because of this mechanism, it can be assumed

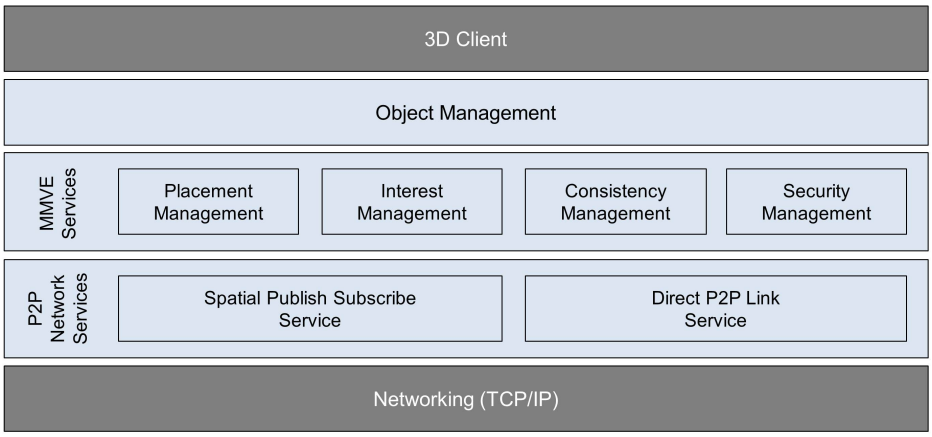


Figure 2.5.: Layer model of the target system

that a superpeer has up-to-date states of all AoIs of peers within its zone. This is important for this work.

The described event propagation mechanism based on spatial influence can be classified as *spatial publish subscribe* (see [Hu09]). AoEs can be interpreted as *spatial publications* and AoIs as *spatial subscriptions*. In the remainder of this work, this terminology is used.

MMVE Software

In order to operate the P2P-MMVE system, each peer runs a special *MMVE software*. Figure 2.5 presents an overview of the architectural layers of the MMVE software of the target system. The components of the architecture are described in detail in the following.

The *3D Client* presents the end users a 3-dimensional graphical view on the virtual environment. The content that is needed to display the virtual environment, for example textures or maps, is installed together with the software on the peers before a peer can join the MMVE system. Additional content is provided after the initial installation of the software via peer-to-peer file sharing. Before a user can log in, the system checks if new content is available. In case new content is found, this content is retrieved from other peers. There is no streaming of content at runtime. The 3D Client also collects user input via keyboard and mouse and translates it into avatar actions.

The *Object Management* is responsible for holding the local copies of objects in the surroundings of the user avatar. Interactions with these objects are handed over from the 3D Client to the Object Management for further processing. Incoming events from other peers over the network are also handed over to the Object Management in order to adjust the state of the local object copies. The Object Management is able to process events that apply an effect to a given object as well as events that apply an effect to objects within a given spatial area. In addition, the Object Management is involved into system tasks regarding objects, for example storing the world state persistently.

The MMVE software includes several *MMVE services* for running the MMVE. The *Placement Management* is responsible for assigning calculation tasks to certain peers, for example the calculation of weather effects in certain areas or the control of non-player characters. The *Interest Magement* is responsible for management of AoIs. The *Consistency Management* provides several consistency concepts for the system. It solves potential conflicts between events, orders incoming events on peers and makes sure that only consistent state changes are propagated and applied to object copies. The *Security Management* protects the system against malicious attacks and cheating.

For propagation of events and communication between the system parts on different peers, the software provides a selection of *P2P network services*. It includes a *Spatial Publish Subscribe Service* that allows to make spatial subscriptions for network messages within a certain area. The subscriptions can include additional filters, for example to subscribe for only a certain message type. In addition, it allows to make spatial publications of network messages to a given area. In case the spatial publication and a spatial subscription and its filters intersect, the network message is delivered to the corresponding peer. In the target system, the Spatial Publish Subscribe Service is used for propagation of events based on AoIs and AoEs. AoIs are registered as spatial subscriptions. Events are propagated to their AoEs as spatial publications. In addition to the Spatial Publish Subscribe Service, the P2P network services include a *Direct P2P Link Service*. In the target system, every peer has a unique peer id. This service allows to send network messages directly to a certain peer based on its peer id. For sending the messages over the network, the P2P network services use TCP/IP.

2.4. Assumptions

This work is based on the following general assumptions:

Time is loosely synchronized between peers - The presented approach makes heavy use of timers. Therefore it is important to have a synchronized time on all peers that participate in the MMVE. Obviously, a strictly synchronized time constitutes the optimal solution. However, the approach works at minimum under the assumption of a loosely synchronized time between peers. For the remainder of this work, it is assumed that time is at minimum loosely synchronized between all peers that participate in the MMVE.

No handling of cheating or provision of security - The presented approach includes distributed calculation of MMVE code on end-user computers. This can be a potential point of attack for cheaters and hackers. Providing specific security concepts and protection against cheating is beyond the scope of this work. As described earlier in this work, the MMVE software of the target system model includes the MMVE service Security Management that protects the system against malicious attacks and cheating. For the remainder of this work, it is assumed that the existing Security Management provides a security concept and protection against cheating.

No handling of consistency and event ordering - The presented approach calculates future state changes on peers and locally creates artificial single events instead of sending single events over the network. The calculation of state changes can potentially lead to a weaker consistency than sending the specific state changes because function calls of the corresponding continuous event can be missed and the actual state of a continuous action can deviate from the calculated state. In addition, consistency has to be provided in case of conflicts between events. Incoming single events from the network and locally calculated artificial single events have to be ordered. The described approach includes a mechanism for detecting missed function calls and for converging continuous events towards a current state. However, providing consistency in case of conflicting events and ordering incoming single events from the network and locally created artificial single events is beyond the scope of this work. As described earlier, the MMVE software of the target system includes the MMVE service Consistency Management. For the remainder of this work, it is assumed that consistency in case of conflicting events and the ordering of events is provided by the existing Consistency Management of the target system.

2.5. Requirements

The presented work has to fulfill the following requirements:

Scalable propagation of continuous actions - The approach aims at a scalable propagation of continuous actions in event-driven P2P-MMVEs with an event propagation via spatial publish subscribe. In the context of this work, enhancement of scalability can be measured by the reduction of the number of messages and bandwidth for propagation of continuous actions over the network. The approach has to be compared to a basic event-driven P2P-MMVE with an event propagation via spatial publish subscribe using single events. In addition, it has to be compared to a state-of-the art MMVE approach that is client/server-based and performs dead reckoning for object movement.

Support of explicit use - The scalable propagation of continuous actions has to be supported for explicit use. Continuous actions do not have to be identified automatically for a scalable propagation. Instead, the scalable propagation of a continuous action can be started explicitly by the MMVE software via a given interface. Changes to running continuous actions that were already propagated or their termination are also triggered explicitly by the MMVE software via given interfaces.

Encapsulation of execution and management processes by the system - After the scalable propagation of a continuous action was started explicitly by the MMVE software, any following processes for calculation and application of state changes as well as for propagating running continuous actions to new peers have to be performed automatically by the system. After a change to an already propagated continuous action or its termination was triggered explicitly by the MMVE software, the system has to distribute the information automatically to all peers that need it.

Handling of peer crashes - Peers can leave the system unintended because their hardware or software breaks down. This has to be considered in the context of the presented approach.

Handling of peer disconnections - Peers can leave the system unintended because of network problems. Peers that leave the system because of disconnections from the network have to be considered in the context of the presented approach.

Handling of overloaded peers - The performance of calculations on peers potentially increases the CPU and memory load. The handling of peers whose CPU and memory get overloaded has to be considered in the context of the presented approach.

3. The Continuous Events Approach

This chapter presents the continuous events approach for scalable propagation of continuous actions in P2P-MMVEs. Section 3.1 gives an introduction into the concept of continuous events. Section 3.2 presents a formal model for continuous events. Finally, Section 3.3 discusses the timing of continuous events and potential use cases for finite and infinite continuous events.

3.1. The Concept of Continuous Events

Section 2.3 described the target system of this work. The target system is an event-driven P2P-MMVE. It propagates user actions based on events and their spatial influence on the virtual environment according to the spatial publish subscribe communication model. The MMVE software of the target system translates user input into avatar actions. It creates events that describe the state changes caused by avatar actions. Each event has an AoE that corresponds to its spatial influence on the virtual environment. The events are propagated to peers based on relevancy. Relevancy is determined by checking for intersections between the AoE and the AoIs of avatars of peers.

There are certain types of user actions in MMVEs that result in multiple subsequent state changes of objects. Figure 3.1 on page 48 illustrates a typical pattern of such a *continuous action*. As result of a single user action, an object is created at location A. In the following, the object moves through the virtual environment towards location B. When the object reaches location B, it influences user avatars and other objects that are located within a given influence area. The object of the continuous action described by Figure 3.1, for example, influences avatars and objects within a circular area that has a given radius and is centered at location B.

Examples for such continuous actions can be found in a wide variety of MMVE use cases. In military simulations, for example, the object can represent a rocket. The rocket is fired off at location A and aims at location B. When the rocket reaches location B, it harms avatars and buildings within a circular influence area. In a

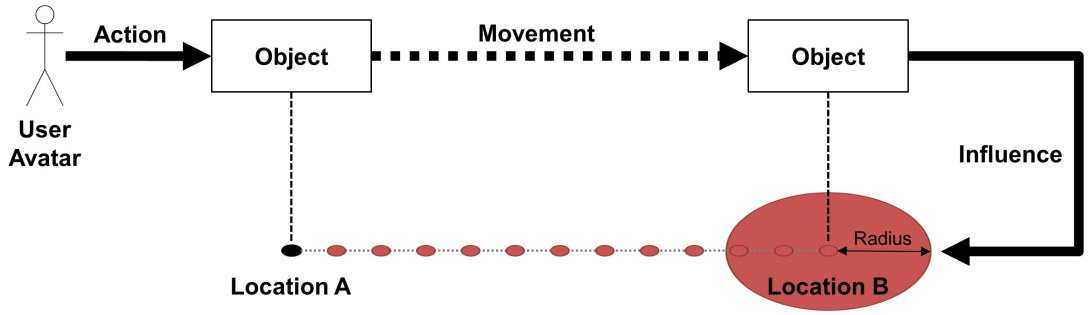


Figure 3.1.: Pattern of a typical continuous action

serious game that simulates a disaster for training purposes, the object can represent a patch of fire. The patch of fire is created at location A. In the following, the patch moves towards location B according to the wind in the virtual environment. Please note that in this use case the influence areas deviate from the areas illustrated by Figure 3.1 because the patch of fire influences avatars and objects on its entire way through the virtual environment instead of avatars and objects within one area at location B. In a business use case, the object can represent a car on an assembly line. The car is put on the assembly line at location A. After that, it moves on the line towards location B. When the car reaches the end of the line at location B, an acoustic signal is triggered that notifies about the arriving of the car. The acoustic signal is heard by all avatars that are located within a given circular area.

In the basic event-driven spatial publish subscribe communication model of the target system, each state change caused by such a continuous action has to be represented by an event with a corresponding AoE. This results in a potentially large number of events that have to be sent over the network for the propagation of continuous actions. The propagation of the continuous action illustrated by Figure 3.1, for example, results in update events about the position changes of the object over time and the final influence at the target location. The update events about the position changes each are propagated to an AoE of a point at the location of the object. The update event about the final influence is propagated to a circular AoE that has the given radius and is centered at the target location.

MMVEs typically consist of a very large number of users, objects and actions. Therefore, the described propagation of continuous actions with multiple single events puts a high burden on the sending side of the network connection of the computer where

a continuous action is performed as well as on the propagation infrastructure. In P2P-MMVEs without a central server infrastructure, the propagation of events is completely performed by user computers that act as peers. Events are propagated directly between the peers. The bandwidth of network connections of users is typically very limited. In addition, the network connections of users are often asymmetric. The connections have a much higher download than upload bandwidth. This results in a potential bottleneck for network traffic at the sending side of the network connections of the peers. Therefore, propagating continuous actions with multiple single events without any optimization can compromise the overall scalability of a P2P-MMVE system because the potentially large number of single events resulting from continuous actions increases network traffic on the sending side. Finding a way to optimize the propagation of continuous actions is crucial for the overall scalability of the system.

MMVE systems usually have a closed set of possible actions that can be performed by users. These actions are designed and implemented during the development of a MMVE. After the MMVE was launched and is available for users, it provides the users with a predefined set of actions. The users are able to make inputs and perform one of the given actions. Usually, they can not add their own actions or perform actions that are not known by the MMVE system. Even in MMVEs that make heavy use of user-generated content, for example Second Life, the set of actions is typically closed. This observation applies to actions that result in a single state change as well as to continuous actions.

Another observation that can be made in the context of continuous actions is that the entire future outcome and all future influences of continuous actions typically can be described at the point in time when the action is performed. At least, a pattern for the future outcome and influences can be described. For the continuous action illustrated by Figure 3.1, for example, the future movement of the object and the final influence at location B is known at the point in time when the user action is performed.

These observations can be used to optimize the propagation of continuous actions. Because the set of user actions is closed and the future outcome can be described at the point in time when a continuous action takes place, all future state changes and influences of a continuous action can be aggregated and propagated to other peers at the point in time when the continuous action is performed. The peer where the continuous action is performed creates a so-called *continuous event*. The continu-

ous event contains a description of the continuous action. Instead of propagating the outcome of the continuous action over time with multiple single events to other peers, the peer only propagates the continuous event once at the point in time when the continuous action is performed. After that, all peers that received the continuous event calculate and apply the following state changes based on the description contained in the continuous event.

To be able to represent a continuous action by a continuous event, an appropriate description of the outcome of the continuous action over time and an appropriate AoE that aggregates the extent of all future influences of the continuous action have to be found. In contrast to single events, a continuous event is created and propagated at the point in time when the corresponding action is performed. To be able to propagate the continuous event to all peers whose avatars can be affected by the continuous event in the future, an appropriate AoE has to contain or approximate all future influence areas of the continuous action. In the following, this work first discusses the spatial modeling of AoEs for continuous events. After that, it discusses the types of informations that have to be contained in a continuous event in order to be able to calculate and apply its outcome over time.

Figure 3.2 on page 51 presents alternatives to model the continuous action illustrated by Figure 3.1 with single or continuous events and corresponding AoEs. The alternative shown on the top of the figure uses a propagation according to the basic event-driven spatial publish subscribe communication model of the target system. For each position change of the object, a single event is created and propagated to an AoE of a point. When location B is reached, an additional single event is created for the influence and propagated to a circular AoE. This results in multiple single events that have to be sent over the network.

The alternative shown at the bottom of the figure uses a propagation based on a continuous event. The AoE of the continuous event covers the extents of all future influences. The extents of the position changes of the object and the final influence are approximated by an aggregated AoE that consists of a line and a circle. The continuous event that contains the description of the future outcome of the continuous action is propagated at the point in time when the continuous action is performed to the aggregated maximum AoE. After that, future state changes caused by the continuous action are calculated and applied by all peers that received the continuous event.

Figure 3.2 further illustrates that, in addition to the propagation according to the ba-

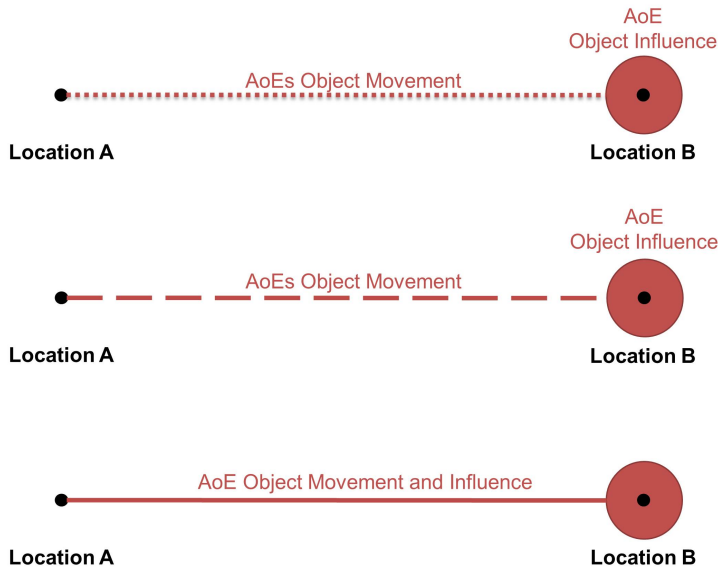


Figure 3.2.: Spatial modeling alternatives for continuous actions

sic event-driven spatial publish subscribe communication model of the target system and the propagation based on a continuous event with a maximum AoE, intermediate modeling alternatives using multiple continuous events and corresponding AoEs are also possible. The alternative in the middle of the figure, for example, represents the continuous action with multiple continuous events. The movement is split into parts and each part is propagated using a continuous event that has an AoE of a line. The final influence is propagated using a single event that has a circular AoE. The use of such an intermediate spatial modeling has the potential to reduce the number of events in comparison to the basic alternative and, at the same time, to provide a higher grade of accuracy in comparison to the alternative with a maximum AoE.

In summary, there are several possible spatial modeling alternatives for continuous events. The alternatives have varying implications on network traffic. On the first hand, using a maximum AoE seems to be the most beneficial alternative because only one continuous event has to be sent over the network. However, using multiple continuous events seems to be a viable alternative as well because it constitutes a trade-off between reduction of network traffic and accuracy. Depending on the characteristics of the MMVE type and use case, reduction of network traffic and accuracy is prioritized differently. In a P2P-MMVE that includes large cities, for example, a lot of users are crowded inside the cities. There is a high user density

at certain places. The users do not pay much attention to the specific actions of all users. In case a continuous action takes place, the outcome has to be propagated to a large number of other peers whose avatars are located nearby. In such a use case, network traffic reduction is prioritized higher than accuracy and one continuous event might be the desired model. In a use case that takes place outside of cities, there are less users and users pay more attention to the specific actions of other users. In such a use case, accuracy is prioritized higher than network traffic reduction and several continuous events or even single events might be the desired model.

In conclusion, there is not one best way to model continuous actions with continuous events and AoEs. Therefore, the continuous events approach aims at providing a generic way to model continuous actions depending on the use case. In order to provide a scalable propagation of continuous actions in P2P-MMVEs, the basic event-driven spatial publish subscribe communication model of the target system is extended with continuous events. The continuous events approach supports the explicit use of varying modeling alternatives. Instead of creating continuous events and a certain spatial modeling automatically, the system for support of continuous events provides a framework for creating and using continuous events explicitly as needed.

In general, three types of informations are needed to calculate and apply the outcome of a continuous action over time: 1) Temporal information, 2) spatial information and 3) information about the effect. In the following, this work discusses these types of information that have to be contained in every continuous event.

In order to calculate changes and influences correctly over time, every continuous event has to contain *temporal information* about when the continuous event starts and ends as well as the interval between every new calculation of state changes and influences. For example, a continuous event representing the continuous action illustrated by Figure 3.1 starts at the point in time when the action takes place and the object is created. It ends at the point in time of the calculation and application of the final influence. The interval determines how often the continuous event performs the calculation of changes and influences in between the start and end. While start and end are clearly defined, the interval provides room for varying alternatives. A small interval results in more calculations. This puts a higher calculation burden on the user computer, but is able to provide a smoother experience for the user because object states are adjusted more often. A large interval results in less calculations. This reduces the calculation burden, but can result in objects that change their

state in an erratic way. Both alternatives seem to be viable. A small interval and a more frequent adjustment of object states is beneficial if enough computing power is available. A larger interval with less calculations is beneficial if it is likely that the participating user computers are not able to handle the calculation load. In conclusion, no specific interval should be defined by the system for continuous event support. Instead, the system provides the possibility to define intervals explicitly as needed.

In order to apply the outcome of the continuous action that is represented by a continuous event to avatars and objects, the continuous event has to contain *spatial information*. Whenever an interval has passed and a continuous event is calculated, the extent of the represented continuous action at that point in time has to be determined in order to decide about the influence of the continuous action on the virtual environment. For example, the extent of the continuous action illustrated by Figure 3.1 can be calculated over time by providing a mathematical function or programming code that returns a point during the object movement and a circle if the target location is reached. In addition to applying the outcome, the spatial information is needed for managing continuous events over time. Because continuous events reside in the system for a certain lifetime, which is defined by start and end, they have to be provided by the system to users that newly join the MMVE during their lifetime. To provide this functionality, the information about the extent of the represented continuous action at a given point in time is needed. More about the management of existing continuous events is given in the next chapter of this work about a system architecture for continuous event support in P2P-MMVEs.

Finally, every continuous event has to contain *information about the effect* of a continuous action over time. Whenever a continuous event is calculated, the system has to determine what has to be applied to the virtual environment or affected objects at that point in time. For example, the continuous event representing the continuous action illustrated by Figure 3.1 has to move the object to the location that is calculated based on the contained spatial information after each interval until the end is reached. When the continuous event is calculated for the last time, it has to move the object to the target location and, in addition, has to apply the influence of the object on the target area. The determination and application of the effect of a continuous event at a given point in time can be performed, for example, based on function code that is called and performs the application of state changes and influences.

The continuous action illustrated by Figure 3.1 involved a single object that changes state over time and results in one final influence. However, continuous actions can also involve multiple objects that change state over time and influence the environment. For example, instead of triggering the movement of a single car on a single assembly line, a continuous action can trigger the movement of multiple cars on multiple assembly lines. Because all cars move according to the same movement pattern, the same function code for calculation of the spatial extent and the effect can be used for all cars. The future state changes and influences of all objects can be propagated in a scalable way by using one common continuous event. To be able to support such complex continuous actions with multiple objects, continuous events have to be conceptually independent from MMVE objects.

In summary, in order to provide a scalable propagation of continuous actions in P2P-MMVEs, the *continuous events approach* extends the event model of the target system with an additional explicit event entity for continuous events. Continuous events represent the future outcome of continuous actions independently from MMVE objects. They reside in the system until their end is reached or until they are terminated explicitly. They carry the information about when a continuous action starts, when it ends, and when a calculation should be performed. A continuous event is able to calculate the spatial extent of the represented continuous action by calling generic function code that returns a spatial shape. The calculation of the effect and the application are also performed by calling generic function code.

Because continuous events are conceptually decoupled from MMVE objects, only carry descriptive information about the continuous action and work based on generic function code, the continuous events approach is able to support a wide variety of modeling alternatives and use cases. One continuous event is able to change the state of a single object as well as a large number of objects over time. This allows to propagate a potentially very large number of state updates with one continuous event. Because generic function code is called, the approach is conceptually able to provide the possibility to add own function code to the system.

Figure 3.3 on page 55 gives on overview of the basic concept of continuous events. Conceptually, the continuous events approach decouples the calculation and application of future changes and influences from MMVE objects. The calculation and application is performed by explicit continuous event entities. Continuous events reside in the system for a given lifetime. They carry the information that is needed for the calculation and application. They call generic function code to calculate the

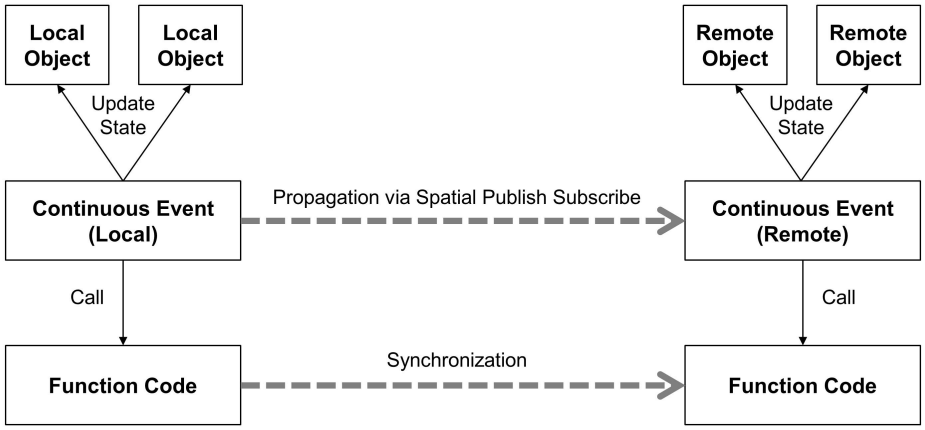


Figure 3.3.: Basic concept of continuous events

future spatial extent and effect of the represented continuous action at a given point in time and to apply the results to MMVE objects. Continuous events are created explicitly by the MMVE software on the peers. After a continuous event was created on a peer, it is propagated to other peers according to the spatial publish subscribe communication model. In the following, the local peer where the continuous event was created and the remote peers that received the continuous event over the network perform the calculation and application over time. To be able to call the same function code on all peers, the function code has to be synchronized between the peers. This can be done either by sending the code included in the continuous event or by synchronizing the code before runtime. This topic is discussed later on in the next chapter of this work about a system architecture for continuous event support in P2P-MMVEs. In the following, this work presents a formal model of continuous events.

3.2. Formal Model of Continuous Events

A continuous event CE can be modeled formally as a 6-tuple (compare [HSS+12]):

$$CE = \{t_0, \delta t, n, f_{AoE}(t), f_E(t), f_{apply}(AoE, E)\}$$

The temporal information of CE is described by t_0, δt and n. t_0 corresponds to the point in time when the continuous action that is represented by CE starts. δt describes the interval between every calculation of CE. n defines the number of

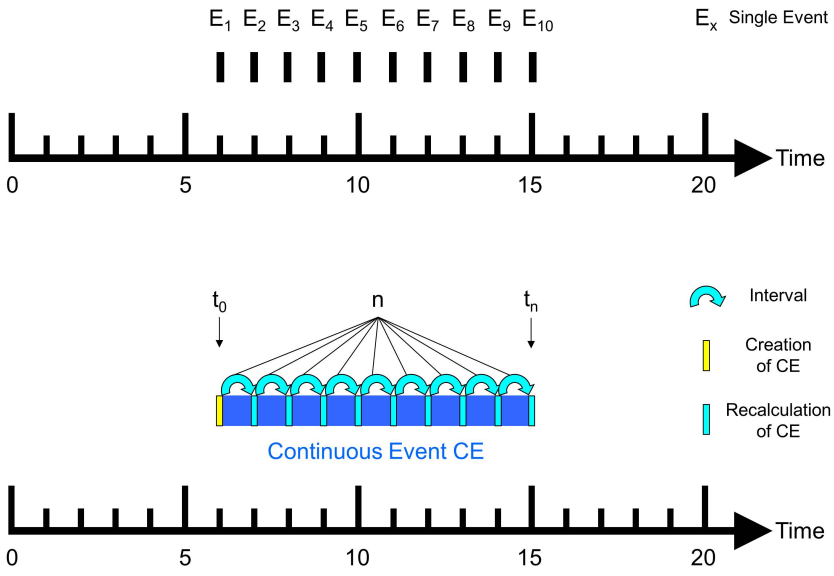


Figure 3.4.: Timing of continuous events

intervals that have to be completed in case CE is not terminated explicitly before n is reached. Please note that the formal model does not explicitly include the point in time when CE ends. The end of CE can be determined in dependency of the included information as $t_0 + n * \delta t$.

The spatial information of CE is described by the function $f_{AoE}(t)$. $f_{AoE}(t)$ calculates the spatial extent AoE of CE at a given point in time t. As described before, the result of $f_{AoE}(t)$ can change over time, for example the function can return an AoE that includes one or multiple spatial structures with a different size or shape.

The information about the effect of CE and the application of the effect are described by the functions $f_E(t)$ and $f_{apply}(AoE, E)$. $f_E(t)$ calculates the specific effect of CE at a given point in time t, for example a certain change of an object property. Similar to $f_{AoE}(t)$, the result can also change over time, for example to increase the value of an object property over time or to shift completely to an other effect with a varying characteristic. $f_{apply}(AoE, E)$ applies the effect locally on a peer based on the results of $f_{AoE}(t)$ and $f_E(t)$.

Figure 3.4 illustrates the timing of continuous events as described by the formal model. Please note that the axes of coordinates for time are labeled with exemplary numbers. Conceptually, the continuous events approach is not correlated to a

certain time unit. At the top, Figure 3.4 illustrates the timing of the propagation of a continuous action using single events. Without any optimization, a continuous action is split into multiple single events that are each propagated at the respective point in time. At the bottom, Figure 3.4 illustrates the timing of the propagation of the same continuous action using a continuous event. The same continuous action can be represented and propagated using one continuous event CE. CE is created and propagated at the point in time t_0. The creation is combined with the first calculation of CE and application of state changes by calling the functions $f_{AoE}(t)$, $f_E(t)$ and $f_{apply}(AoE, E)$. After that, the peer where CE was created and the peers that received CE recalculate CE n times based on the given interval length δt. In order to recalculate CE over time, the functions $f_{AoE}(t)$, $f_E(t)$ and $f_{apply}(AoE, E)$ are called repeatedly. The end of CE t_n is reached if n intervals have passed.

3.3. Timing Alternatives: Finite and Infinite Continuous Events

The presented formal model and timing of continuous events assumed that the end of continuous events can be defined at the point in time of their creation. For each continuous event there is a given number of intervals n. The end of the continuous event is reached if n intervals have passed. In other words, the continuous events that were discussed so far are *finite*.

However, not all continuous actions have a predefined end. In addition, it might also be desirable to have continuous events without a predefined end from the perspective of MMVE design. A MMVE system for simulation of distasters, for example, might want to create a fire patch without deciding beforehand about the point in time when the fire has to be put out. Instead, it might want to create the fire by using a continuous event, control the continuous event over time and terminate the continuous event explicitly in order to put the fire off.

To support such use case scenarios, the continuous events approach includes *infinite* continuous events in addition to finite continuous events. Conceptually, infinite continuous events have a clearly defined start, analogical to finite continuous events. The start of infinite continuous events is marked by the point in time t_0 when the continuous event is created, propagated and calculated for the first time. Infinite continuous events also have a given interval δt that describes the time span between the recalculations of the continuous event. In contrast to finite continuous events,

infinite continuous events do not have a given number of intervals n. Instead of performing the recalculation of the continuous event n times, the recalculation of an infinite continuous event is performed infinite times.

The introduction of infinite continuous events allows to support use cases with potentially very complex continuous events. Weather effects in MMVE systems, for example, typically are not very dynamic. In order to improve the scalability of the MMVE system, effects like rain are scripted. Instead of presenting rain clouds as objects in the virtual environments and applying rain effects based on the position of these clouds, rain is available as a scripted event on user computers. The scripted event is started from time to time in order to provide the users with the experience of changing weather. Infinite continuous events have the potential to enable the presentation of sophisticated rain clouds in P2P-MMVEs in a scalable way. In order to support this, a peer can be assigned with the task of controlling the weather effects for a given area. In the following, this peer calculates conditions like wind direction and speed for the assigned area. In order to present clouds and to apply the impact of rain on other peers without creating a lot of network messages, this peer can create and sent one continuous event. This continuous event has an AoE that consists of multiple spatial shapes. Each of these shapes represents the position of a cloud object and the influence of the rain of this cloud on the environment. After the creation and initial propagation, the continuous event calculates up-to-date positions for all cloud objects and applies the rain effect over time according to the wind direction and speed. Figure 3.5 on page 59 illustrates the described example.

Infinite continuous events that reside in the system can be controlled explicitly. In case the wind direction and speed should change, for example, the peer that controls the weather sends another message to all peers that have a copy of the continuous event. In order to save network bandwidth, only the adjusted parameters for the functions need to be sent instead of creating and sending a new continuous event. The calculation of the cloud objects and their influence then can be adjusted on the peers based on the received information. In case the rain should be stopped, for example, the peer that controls the weather sends a message for termination of the continuous event copies on all peers.

Because the continuous events approach uses generic function code, it is able to support even more complex alternatives of the described use case. For example, the functions for calculating the positions of clouds and their influence over time could consider the dilution of clouds in the air. In addition to changing the position, the

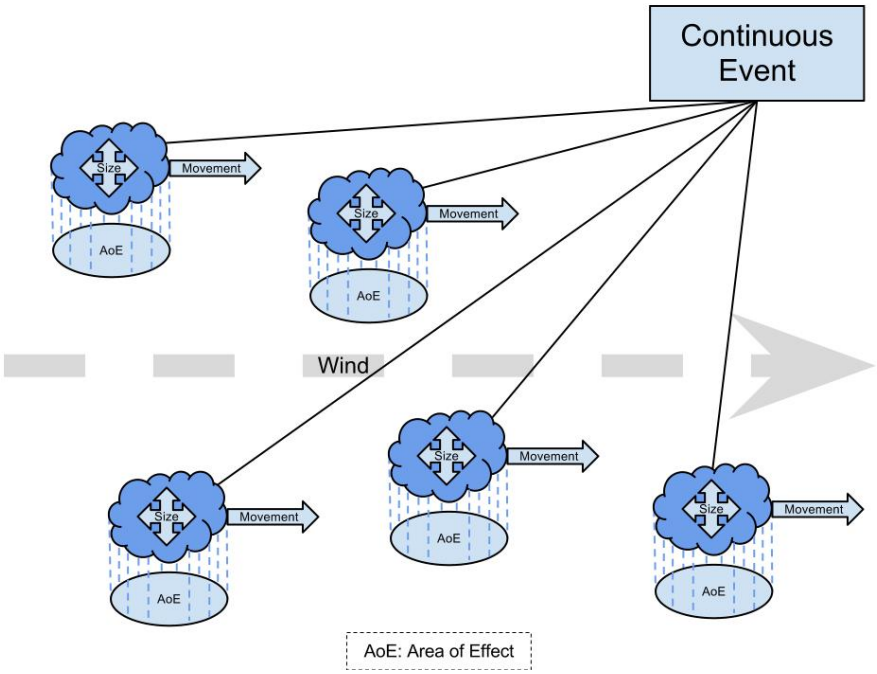


Figure 3.5.: Complex use case example for continuous events

functions could shrink the size of the cloud objects and their influence over time. Another alternative of the described use case with a higher grade of complexity could be the inclusion of the reactions of non-player characters to the rain clouds. For example, the functions called by the continuous event could force all non-player characters within the influence area to open umbrellas. Assuming the function code that is needed to perform the actions in the virtual environment is available on all peers, no additional network bandwidth is needed at runtime to support such alternatives of the use case with a higher grade of complexity.

From a conceptual point of view, there are use cases that can be represented by both timing alternatives. In Section 3.1, this work presented the example of a continuous action that creates an object and moves the object from location A to location B. Finally, the object influences other objects in the surroundings of location B according to a given influence area (compare Figure 3.1 on page 48). Such a continuous action can be represented by an infinite continuous event as well. A propagation of the continuous action based on an infinite continuous event can be described as follows: The infinite version of the corresponding continuous event is initially propagated to an AoE of a point at location A. After that, the continuous event calculates the new positions of the object and moves the object into the direction of location B without including the information about when to stop the

movement. At the point in time when the object arrives at location B, the peer where the continuous event was initially created terminates the continuous event explicitly. After that, it propagates the final influence with an additional single event. Using a finite continuous event seems to be the better alternative for the given use case because no additional single event has to be sent and, as a result, less network traffic is created. Nevertheless, it is possible to use an infinite continuous event as well.

On the other hand, the example use case for infinite continuous events that was presented in this section can also be represented by a finite continuous event as well. Instead of terminating the movement of clouds and the application of rain effect explicitly, the continuous event can include a given number of intervals after that it is terminated automatically. The number of intervals can be chosen, for example, so that the cloud objects are created and start to move at one border of a given area in the virtual environment and are terminated automatically when they reach the other border of the area. Using a finite continuous event for this use case has the potential to reduce the network traffic for propagation because no explicit termination message has to be sent. However, a finite continuous event provides less flexibility compared to the alternative with the infinite continuous event because the end has to be predefined and an appropriate number of intervals has to be found.

The described observations show that a decision about the timing alternative that should be used for a certain use case can not be made in general. The selection of a specific timing alternative has implications on flexibility and resulting network traffic. A decision about a timing alternative can only be made in dependency of the use case. Therefore, the continuous events approach supports the use of both timing alternatives. This has to be considered for the design of algorithms and a system architecture for continuous event support.

This chapter presented the basics of the continuous events approach. The approach enables a scalable propagation of very complex continuous actions and supports a variety of use cases in P2P-MMVEs. This chapter emphasized the conceptual aspects of the approach. It introduced the concept of continuous events, gave a formal model and discussed timing alternatives and potential use cases. The next chapter puts an emphasis on the realization of the continuous events approach in the context of P2P-MMVEs. It describes the design of algorithms and a system architecture for support of continuous events in P2P-MMVEs.

4. A System Architecture for Continuous Event Support in P2P-MMVEs

This chapter presents a system architecture that supports continuous events in the context of the target system model of a P2P-MMVE with zones and superpeers. Section 4.1 identifies the functionalities that have to be provided for continuous event support, discusses the overall design of the system architecture and gives an overview of the system architecture and its components. Section 4.2 describes the basic support of continuous events by the architecture and gives algorithms for basic support. Finally, Section 4.3 presents and discusses several extensions for the system architecture.

4.1. Architecture Design

Based on the requirements that were presented in Section 2.5 and the description of the continuous events approach in the previous chapter, the following three main functionalities for continuous event support can be identified:

1. *Explicit use of continuous events* - In the previous chapter, it was shown that there is no modeling for continuous events that fits all use cases and continuous actions. In addition, infinite continuous events have to be modified and terminated explicitly and no general decision can be made about a timing alternative that fits all use cases and types of continuous actions. The system achitecture aims at providing continuous event support for P2P-MMVEs in a generic way without focusing on a certain use case or type of continuous action. Therefore, the system architecture has to support the explicit use of continuous events by the peers. It has to provide an interface for explicit creation of new continuous events as well as explicit modification and termination of existing continuous events.
2. *Execution of continuous events on target peers* - For support of the continuous events approach, incoming continuous events have to be executed automati-

cally by the peers. In order to support this, the system architecture has to provide an automatic execution mechanism that includes the automatic calculation of continuous events and application of their effect over time. In case no explicit termination takes place, continuous events have to be terminated automatically after the given lifetime is over and the end is reached.

3. *Management of existing continuous events* - For support of the continuous events approach, existing continuous events have to be managed automatically by the system. The available AoI subscriptions in the system can change because of peers that leave or join the MMVE or because of avatar movement. In addition, the AoEs of existing continuous events can change location and size over a continuous event's life time based on the functions that are called over time. Changing AoI subscriptions and AoEs can lead to new intersections between the AoEs of existing continuous events and AoIs of peers that do not execute these continuous events yet. Therefore, the system architecture has to include a mechanism that detects these new intersections and provides the peers with up-to-date copies of existing continuous events. In addition, the system architecture has to provide a mechanism to address all available copies of a continuous event in the system in order to be able to modify or terminate all copies of a specific continuous event. Analogical to functionality 2, in case no explicit termination takes place, the management of continuous events has to be stopped automatically by the system after the given lifetime is over and the end is reached.

For full support of the continuous events approach, these three main functionalities have to be provided by the system architecture. Functionalities 1 and 2 have to be provided on every peer because every peer has to be able to use continuous events explicitly and to execute continuous events locally. Functionality 3 can also be provided on every peer. However, it does not need to be provided on each peer. Continuous events can also be managed by only a subset of peers or even in a centralized way by a single peer. In case a single peer manages the continuous events for the whole virtual environment, this peer has to calculate the up-to-date state of all existing continuous events. In addition, the peer has to be provided with all AoI subscriptions to be able to detect new intersections between AoEs of existing continuous events and AoI subscriptions. In order to be able to perform these tasks, the peer needs very strong hardware and networking capabilities and has to be selected by the system based on these criteria. In case a subset of peers manages continuous

events, the burden that is put on the computational and network resources can be distributed over multiple machines. To enable continuous event management by a subset of peers, algorithms for selecting appropriate peers, for providing each of these peers with a subset of AoIs, for assigning continuous events to peers and for coordinating between the managing peers are needed. The alternative of using all peers for continuous event management has the potentially best distribution of load because all available machines are used. Analogical to the alternative using a subset of peers, this alternative also needs algorithms for selecting appropriate peers, for providing each of these peers with a subset of AoIs, for assigning continuous events to peers and for coordinating between the managing peers. A comprehensive discussion of the alternatives for continuous event management on the participating peers is given later on in this chapter. In the following, this work introduces the architectural components and discusses their placement on peers.

The system architecture for support of the continuous events approach distributes the provision of the identified functionalities over architectural components as follows: Functionality 1 is provided by the so-called *Continuous Event Controller* (CEC). The CEC provides interfaces for explicit use of continuous events and controls the outgoing and incoming information flow. The provision of functionality 2 and the provision of functionality 3 both have similar needs. For example, for both functionalities the calculation of continuous events over time has to be performed. However, since continuous event management can also be performed by a subset of peers or by a single peer and the varying alternatives have direct implications on the overall system performance, the execution and management of continuous events is split into two architectural components. This allows to execute continuous events on all peers while managing continuous events by all, by a subset or by a single peer. The execution of continuous events is provided by the so-called *Continuous Event Executor* (CEE). The management of continuous events is provided by the so-called *Continuous Event Manager* (CEM). In the following, the CEC, CEE and CEM are described in more detail. The collaboration of the architectural components is explained later on in Section 4.2.

A CEC is run on all peers that participate in the MMVE. It provides interfaces for the explicit use of continuous events by the MMVE software on the peers. Continuous events can be started, modified and terminated via the CEC. In addition to providing interfaces, the CEC is responsible for handling the outgoing and incoming information flow to and from the network. Locally created continuous events are

propagated by the CEC to other peers over the network via spatial publish subscribe. Incoming continuous events are received by the CEC. The CEC hands them over to the CEE or CEM for future performance of their respective tasks. In case CEEs or CEMs that are located on different peers have to communicate with each other over the network, this is handled by the CEC as well.

A CEE is also run on all participating peers. The CEE receives continuous events from the CEC that were either received from other peers or locally created on the peer. The CEE is then responsible for calculating the continuous events over time and for applying the state changes locally to the MMVE objects. For the calculation of continuous events and application of state changes, the CEE calls generic function code in given intervals as described in Chapter 3.

In the context of the target system model as described in Section 2.3, two potential alternatives for applying the state changes can be identified. The called function code can apply the state changes directly to the MMVE objects, for example by modifying the object properties. Alternatively, the state changes can be translated into artificial single events that are created by the called function code and are conceptually identical to incoming single events from the network services of the MMVE software. Because these artificial single events are conceptually identical to incoming events from the network services, the existing services for running the MMVE can be reused. Consistency and security can be provided by the respective services and the existing object management can be used for applying the state changes described by the artificial single events to the MMVE objects.

A direct application of state changes demands additional interfaces from the target system that allow the direct manipulation of MMVE objects without using the existing MMVE services of the target system, such as Consistency Management or Object Management. It is a challenging task to provide such interfaces because the continuous events approach aims at supporting generic function code. In addition, the functionalities of the existing MMVE services have to be provided by the continuous events when the functions are called. This creates a high complexity for the system architecture for continuous event support and leads to redundancy in the overall context of the target system. The functionalities also can not be provided without coordination with the existing services because continuous events and these services provide the same functionalities for the same MMVE objects. It is a challenging task to realize this for generic function code.

In conclusion, applying the state changes directly is not a viable solution in the context of the target system. Therefore, the CEE is designed to create artificial single events. When executing continuous events, the function code is called and based on the results one or multiple artificial single events are created that are conceptually equal to incoming single events from the network services. The existing MMVE services, such as Consistency and Security Management, are used to perform their tasks for incoming single events from the network services and for locally created artificial single events. The existing Object Management is responsible for applying the state changes described by the single events to MMVE objects.

A CEM is responsible for managing a set of existing continuous events over time. As mentioned before, a CEM can be run at minimum on one peer and at maximum on all peers. In case of a single CEM, the managed set consists of all existing continuous events in the MMVE system. In case of multiple CEMs, the set of all existing continuous events has to be split into subsets and these subsets have to be assigned to the CEMs. In addition, the CEMs each have to be provided with all AoI subscriptions that are needed to be able to detect new intersections between the AoEs of their managed continuous events and the AoIs of peers in the surroundings.

In order to find an appropriate design for the CEM, five aspects have to be considered and are discussed in the following: The extent of the calculations that have to be performed by the CEM (1), the number of peers with CEMs (2), the placement of CEMs on peers (3), the provision of AoI subscriptions to CEMs in the context of the target system (4), the addressing of available continuous event copies in the system for modification and termination (5).

Concerning the extent of the calculations that have to be performed by the CEM (1), at minimum the CEM needs to calculate the AoE of each managed continuous event because this information is needed for the detection of new intersections between the AoE and AoI subscriptions of peers. In case only the AoE of a continuous event is calculated by the CEM, there is no up-to-date state of the full continuous event available at the CEM. Baring any unforeseen circumstances such as peer crashes, an up-to-date state of the full continuous event is at least available at the CEE of the peer where the continuous event was created. In addition, an up-to-date state is available at the CEEs of all peers that received the continuous event. In case the CEM only calculates the AoE of each managed continuous event and detects new intersections with AoIs, an up-to-date state of the full continuous event either has to be retrieved from one of the CEEs and sent to the peer with a newly intersecting

AoI or the direct sending of the full state from one of the CEEs to this peer has to be triggered by the CEM. At maximum, the CEM performs a full calculation of the continuous event. All functions are called, analogical to the execution of continuous events by the CEE. In contrast to the execution of continuous events, no artificial single events have to be created and applied. In case the CEM performs a full calculation, peers whose AoI newly intersects the AoE of a managed continuous event can be provided directly by the CEM with an up-to-date state of the continuous event.

In comparison, a minimum calculation puts a lower calculation burden on the hardware of the peers that run a CEM because less functions have to be called. However, the retrieval of up-to-date states of continuous events from one of the CEEs or the triggering of direct sending of states leads to additional network overhead. On the other hand, a full calculation puts a higher calculation burden on the hardware of the peers that run a CEM, but does not need additional network bandwidth for retrieving states or triggering the direct sending of states. Because the continuous events approach aims at reducing network traffic, the CEM is designed to perform full calculations of the managed continuous events. The CEM provides peers with newly intersecting AoIs directly with continuous event copies that have an up-to-date state. The additional calculation burden can, for example, be handled by assigning CEMs to peers with strong hardware capabilities and by making sure that there are always enough CEMs in the system to allow a full calculation of all existing continuous events in the system without overloading a certain peer.

Concerning the aspect of finding an appropriate number of peers with CEMs (2), a CEM can be run on one peer, on a subset of peers or on all peers. In case a single peer manages the continuous events for the whole MMVE system, this peer has to calculate a potentially huge number of continuous events and intersections. In addition, the peer has to be provided with the AoI subscriptions of all peers. Because of the huge number of users and objects in a MMVE, this overloads the hardware and networking capabilities of a single user computer. Therefore, a fully centralized management of continuous events is not a viable option in the context of a pure P2P-MMVE that is run solely by user computers. The task of managing continuous events has to be distributed over a selected subset of peers or over all peers. The system architecture for continuous event support is designed to use multiple CEMs and the set of all existing continuous events in the MMVE system is split and distributed over the CEMs.

Placing a CEM on every peer has the potentially best distribution of load because the task of managing continuous events is performed by all user computers and, in theory, uses all available resources of peers in the system. However, although the use of such a fully decentralized continuous event management distributes load over a huge number of peers, certain peers can still get overloaded. Not all participating peers in a P2P-MMVE are strong enough to be used for additional system tasks. The hardware and networking capabilities of some user computers can be weak and barely strong enough to participate in the MMVE. On the other hand, placing CEMs on only a selected subset of peers allows the system to select appropriate peers with strong hardware and networking capabilities. Such peers are less likely to be overloaded and weak user computers are disburdened from performing system-related tasks. In conclusion, using a subset of strong peers has advantages over using all peers. Therefore, the system architecture is designed to place CEMs on a selected subset of peers.

The use of multiple CEMs implies the need for a criterion and mechanism for placing CEMs on peers (3). Concerning a potential criterion for placing CEMs and assigning continuous events to CEMs, two alternatives come to mind. The placement of CEMs and assignment of continuous events can be performed with or without spatial awareness. An approach without spatial awareness selects a certain number of strong peers based on a given algorithm that measures the hardware and networking capabilities of the peers. CEMs are run on these peers and continuous events are assigned to these CEMs in a balanced way. This makes sure that load is distributed and all peers manage a similar number of continuous events.

In contrast, an approach with spatial awareness tesselates the virtual environment spatially into zones. For each zone a peer with strong hardware and networking capabilities is selected. Each of these peers runs a CEM. The CEM is responsible for managing the existing continuous events within the given zone of the virtual environment. Continuous events are assigned based on their AoEs. Each CEM manages all continuous events whose AoEs intersect its zone.

An approach without spatial awareness is able to balance the load between the peers with CEMs. However, a potentially huge network overhead is created. Without a spatial distribution of continuous events to CEMs, the AoEs of continuous events that are managed by a specific CEM can be potentially located anywhere in the virtual environment. As a result, all CEMs have to be provided with all AoIs in the system in order to be able to detect new intersections between the AoEs of man-

aged continuous events and the AoIs in a reliable way. On the other hand, using an aproach with spatial awareness distributes the continuous events based on their AoEs to CEMs. CEMs only need to be provided with the AoI subscriptions for their zone. As a result, less network traffic is created. In conclusion, an approach with spatial awareness has advantages over an approach without spatial awareness concerning network traffic. Because the continuous events approach aims at reducing network traffic, the system architecture is designed based on an approach with spatial awareness.

In the context of the given target system, an approach with spatial awareness can be realized in a straightforward way. The existing zone tesselation and superpeers of the target system can be reused for continuous event management. Because the superpeers of the target system are selected based on their hardware and networking capabilities, they are unlikely to be overloaded by the additional task of managing continuous events for their zone. A CEM is placed at each superpeer and the continuous events are assigned to the CEMs based on the intersections of their AoEs with the existing zones.

A solution concerning (4) can also be derived from the target system. Because the existing superpeers of the target system act as backup interest matchers for their zones, all AoI subscriptions of a zone are available at the superpeer of the zone. In order to be able to detect new intersections between the AoEs of existing continuous events and AoI subscriptions, the available AoI subscriptions can be used without sending additional network messages because the CEMs are placed at the existing superpeers.

Please note that continuous events can have very large AoEs or AoEs that consist of multiple shapes. As a result, such AoEs can intersect multiple zones. In case continuous events are assigned to CEMs based on their AoEs, this can result in continuous events that have to be managed by multiple CEMs. This creates a need for coordination between these CEMs. Mechanisms for coordination between CEMs are discussed in more detail later on in Section 4.3.

Concerning (5), in order to be able to address all peers that hold a copy of a specific continuous event, the CEM has to maintain informations about which peer received which continuous event for execution. In case a continuous event has to be modified or terminated, all peers with copies can then be addressed based on the stored informations. Therefore, the CEM holds a so-called *receiver list* for each managed continuous event. After a peer received a continuous event for execution, it registers

with the CEM of the superpeer of its zone as a receiver of this continuous event. This registration is updated by the peer in case of changes, for example in case the peer changes its zone. By holding receiver lists for the managed continuous events, the CEM is able to sent modification or termination messages to all peers that execute a copy of a specific continuous event. More about the information flow for modification and termination of existing continuous events is given later on in Section 4.2.

Several additional functionalities can be derived from the described design of the CEC, CEE and CEM that are needed by the three architectural components to provide their functionalities. These additional functionalities are described in the following. Please note that in order to make a clear distinction between all functionalities the earlier enumeration is continued. Therefore, the numbering of the additional functionalities starts with 4.

The following additional functionalities can be derived:

4. *Provision of generic function code on the peers* - The CEE and CEM call generic function code to execute or manage continuous events. The function code has to be available on all peers. In addition, the insertion of additional function code to the system has to be supported.
5. *Use of timers for continuous event calculation* - The CEE and CEM call function code in given intervals. A timer functionality or timer service is needed for this.
6. *Storage of continuous event states between function calls* - The state of a continuous event has to be stored by the CEE and CEM between the function calls. A functionality for storing continuous events is needed.
7. *Propagation of network messages via spatial publish subscribe* - The CEC has to be able to propagate newly created continuous events to other peers based on a spatial publication corresponding to the AoE of the continuous event.
8. *Sending of network messages to specific peers* - The CEC has to be able to send network messages directly to a given peer, for example to send modification or termination messages to the peers that are contained in the receiver list of a continuous event.

Concerning functionality 4, three potential alternatives come to mind. The function code that is called by a continuous event can either 1) be sent included in the

continuous event to all target peers at runtime. Or 2) it can be synchronized between all peers in the system before runtime. At runtime only function ids are included in the continuous event and the code is retrieved and called based on the ids. Or 3) function code can be synchronized before runtime. At runtime function ids and the values of function parameters are included in the continuous event. The code is retrieved and called based on the ids and the parameter values.

In comparison to the other alternatives, the alternative of sending the function code included in the continuous event at runtime provides the most flexibility. Code can be created and sent at runtime. That allows a very flexible creation of continuous events. The downside of this alternative is that the payload of network messages including code is potentially very large. This can overload the upload bandwidth of the peers because the upload bandwidth of the Internet access of end-users typically is very limited. In addition, providing security and protection against cheating for this alternative is a major challenge. Attackers or cheaters can create code, induce the code into the system at runtime and propagate the code to other peers. The continuous events then would act as carriers of potentially malicious code and call functions that harm the system or cheat on other users in the MMVE.

The alternative of synchronizing code before runtime and sending function ids included in the continuous event is the least flexible alternative, but the potentially most scalable alternative concerning network bandwidth. The function code can be distributed between peers before the MMVE is run, for example by downloading the function code from other peers before the user logs in. File and content sharing is a standard use case of peer-to-peer networks and sophisticated algorithms and protocols for file sharing exist. In addition, as mentioned before, the set of available continuous actions of an MMVE is usually closed and defined when designing an MMVE. Therefore, the code for available actions can be written before publishing the MMVE and be included into the MMVE software. Additional code that is added later on by the MMVE provider then can be distributed via peer-to-peer before the user logs in. Similar mechanisms are already used by existing commercial MMVE systems such as World of Warcraft for distributing content updates over the user computers. Compared to the alternative of sending function code at runtime, the alternative that uses function ids results in smaller payloads for network messages because only the function ids have to be included in the continuous event. However, the lack of flexibility can be a huge problem, especially concerning the function code for calculating AoEs. Without any additional parameters, a huge number of func-

tions has to be written and stored in order to provide all potential AoE shapes. In the previously described example of an object that moves from location A to location B as the outcome of a continuous action, the corresponding continuous event, for example, always has the shape of a line with the same length. But depending on the position and direction of the user avatar at the point in time when the continuous action is performed, the starting and ending points of the lines that represent the AoEs of the continuous events can have varying coordinates. The function for calculating the AoE needs at least the position and direction to calculate the line. In conclusion, although the alternative of sending only function ids has clear advantages concerning network traffic over the alternative of sending function code, it is not a viable option because the calculation of AoE shapes can only be supported in a very limited way.

The alternative of sending function ids and additional parameter values for the functions is able to overcome the limitations of the other alternatives. This alternative avoids the large payload and the security and cheating concerns of the alternative that sends code. In addition, it offers more flexibility than the alternative that sends only function ids. This comes at the cost of a slightly larger payload that is needed for sending the additional parameter values. For example, the continuous event that creates the object and moves it from location A to location B can include the id of a function that calculates a line of a given length. In addition to the id of this function, the starting position and direction can be sent as additional parameter values. Please note that there is a lot of flexibility. For example, there can also be a function that calculates a line based on a position, direction and length. In addition to the described example, such as function can also be used for other use cases that involve lines as AoEs. However, the given length has to be sent as an additional parameter value. The higher flexibility comes at the cost of a higher network traffic.

In summary, sending function ids and additional parameter values constitutes the best trade-off between potential payload size and flexibility. Therefore, the system architecture is designed according to this alternative. It includes a so-called *Code Repository Service* (CR). A CR is run on every peer. Function code can be added to the CR of a peer and is synchronized between the CRs of the peers before runtime. If new function code is added, the CR assigns a unique id for the function. The CR includes an interface for the CEE and CEM that allows to retrieve the code for a given function id. In case a continuous event has to be calculated, the CEE and CEM retrieve and call the function code based on the function ids that are

contained in the continuous event. To make sure that the available function code is up-to-date, the CR searches for newly available function code in the system after the MMVE software is started on a peer by a user. If new function code is detected, it is downloaded from the CRs of other peers. The user can not log in before the search and a potential download of code was completed.

In order to execute or manage continuous events, the CEE and CEM need to make use of timers to trigger the calling of function code (functionality 5). To provide this functionality, the system architecture includes a *Timer Service* (TS). A TS is run on every peer and provides a service for registration of timers for CEE and CEM. A performant timer service is available in most programming languages and can be reused when implementing the described system architecture. Therefore, no further discussion about the design of a timer service is given here.

During the intervals between function calls, the state of continuous events has to be stored by the CEE and CEM (functionality 6). Two potential design alternatives concerning the storage of continuous event states come to mind. The storage can either be integrated into the design of the CEE and CEM and be performed by CEE and CEM independently from each other. Or the storage can be provided for both architectural components by an additional component or service.

Including the storage into the design of the CEE and CEM and storing the states independently from each other facilitates the use of separate storage concepts for each architectural component. This is beneficial, for example, if one of the components needs fast access to the stored states but not necessarily a high grade of persistency while the other component has different requirements. On the downside, a design that includes the storage functionality into the CEE and CEM potentially results in redundancy when implementing the system architecture in case both architectural components have similar requirements.

Providing the storage by an additional component or service has the benefit of reducing code redundancy when implementing the system architecture. The storage service can be optimized towards providing a scalable storage while CEE and CEM can be optimized towards executing and managing continuous events. In addition, an additional service for storage can also provide several storage concepts, similar to the design alternative that includes storage into the design of the CEE and CEM.

In conclusion, the design alternative with an additional storage service has less potential redundancy than the design alternative that includes storage into the CEE and CEM. At the same time, it is also able to provide different storage concepts for

each component if needed. Therefore, the system architecture includes a so-called *Continuous Event Storage Service* (CES). A CES is run on every peer and provides an interface for the CEE and CEM that allows to store and retrieve the state of continuous events.

Concerning a storage concept for the CES, two basic alternatives come to mind. Continuous event states can either be stored transiently in memory or be written persistently to the hard disk. Storing the states in memory allows a faster access to the states than writing and reading the states to and from the hard disk. On the downside, the states can be lost in case the MMVE software on a peer crashes. Writing and reading the states to and from the hard disk has potentially slower access times, but states can be read from the hard disk after a crash of the MMVE software. In addition to these two basic alternatives, a hybrid concept is also possible. A hybrid concept stores continuous event states in memory for fast access and writes the states additionally to the hard disk.

When deciding about a storage concept, it has to be discussed if it is beneficial at all to write continuous event states to the hard disk. Restarting the MMVE application after a crash usually takes a certain amount of time. If continuous event states are written to the disk, the stored states can deviate strongly from the actual system state after the MMVE application is restarted. Therefore, having a persistent copy of continuous event states on the hard disk does not bring a huge benefit. As will be described later on in Section 4.3, after a system crash up-to-date states can be retrieved from other peers in the system instead of reading them from the hard disk. On the other hand, having fast access to continuous event states is a huge factor for scalability of the continuous events approach. Whenever a continuous event is recalculated, the previous state has to be retrieved. Then the functions are called. After that, the new state has to be stored. Having fast access times to the continuous event state increases the performance of the continuous events approach. The benefits of fast access times clearly overweigh the benefits of having a persistent storage. Therefore, the CES stores continuous event states in memory.

Concerning functionalities 7 and 8, the system architecture for support of continuous events can use the existing peer-to-peer network services from the target system as described in Section 2.3. The Spatial Publish Subscribe Service provides the CEC with the possibility to propagate network messages to other peers via spatial publish subscribe. In addition, the management of spatial subscriptions is also performed by the existing Spatial Publish Subscribe Service. Existing AoI subscriptions for a

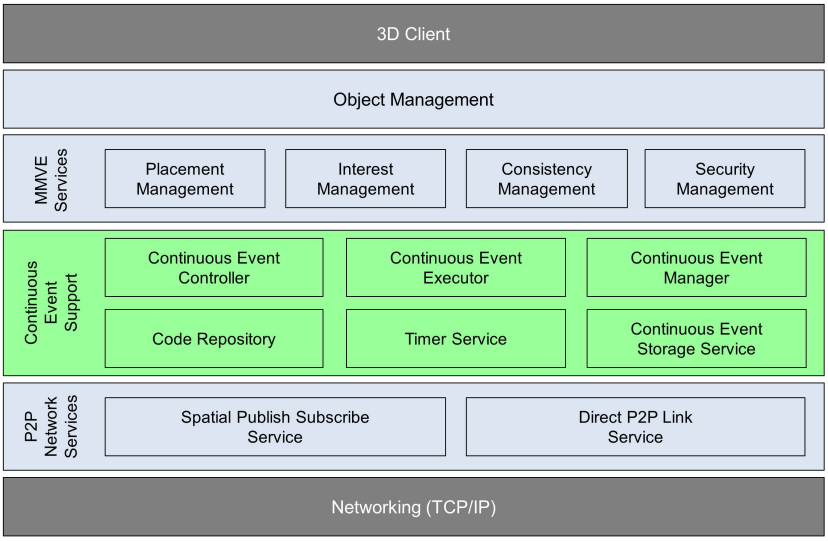


Figure 4.1.: Extended layer model of the target system

zone can be retrieved by the CEM from the Spatial Publish Subscribe Service via an interface. The Direct P2P Link Service provides the CEC with the possibility to send network messages directly to specific peers. Each peer has a unique peer id in the system. The peer id is assigned by the Direct P2P Link Service. Messages to specific peers can be sent by addressing them based on the peer ids.

To conclude this section about the design and components of the architecture for support of continuous events in P2P-MMVEs, Figure 4.1 illustrates the integration of the presented architecture into the overall architecture of the target system. The layer model of the target system that was presented earlier in Section 2.3 (compare Figure 2.5 on page 42) is extended for continuous event support. Another layer is added that includes the architectural components for support of continuous events.

4.2. Basic Continuous Event Support

The previous section presented the overall design and components of the system architecture for continuous event support. This section describes the collaboration of the architectural components and explains in detail how continuous event support is provided. It focuses on basic continuous event support within a zone of the P2P-MMVE. Continuous event support for multiple zones is discussed later on in Section 4.3.

In the remainder of this section, the system behavior and algorithms for the fol-

lowing use cases are described: MMVE start (Subsection 4.2.1), joining of a new peer (Subsection 4.2.2), leaving of a peer (Subsection 4.2.3), start and execution of continuous events (Subsection 4.2.4), management of existing continuous events (Subsection 4.2.5), modification and termination of existing continuous events (Subsection 4.2.6).

Please note that the descriptions and algorithms in this section and the next section refer to the architectural components based on the abbreviations that were introduced in the previous section. The used MMVE and P2P network services from the target system are abbreviated as follows: Interest Management (IM), Spatial Publish Subscribe Service (SPS), Direct P2P Link Service (DL). The given pseudocode uses the conventions for pseudocode from the textbook *Introduction to Algorithms* by Cormen et al. (see [CLRS01, p.19-p.20]).

4.2.1. MMVE Start

According to the target system model that was described in Section 2.3 the system behavior at the start of the MMVE can be described as follows: The target system initially tesselates the virtual environment into zones based on a 2-dimensional projection of the space. For each zone, a superpeer is selected from the set of all available peers in the system. These superpeers perform additional system tasks for their zones. The system tasks include the management of AoI subscriptions and the matching of AoI subscriptions and AoE publications for the provision of the SPS. Therefore, a peer is provided with the AoI subscriptions of all peers within the zone after it is selected as a superpeer and assigned to a zone.

This system behavior is extended for continuous event support as follows: After a superpeer was selected and assigned to a zone, a CEM is started on this peer. All CECs of the peers that belong to the zone managed by the superpeer are provided with the peer id of the superpeer to make sure that the CECs can address network messages directly to the superpeer for continuous event management. In order to receive all continuous event publications whose AoEs intersect the zone, the CEC of the superpeer uses the SPS to perform an additional spatial subscription that corresponds to the shape of the zone. This subscription includes a filter that makes sure that the CEC is only provided with continuous event messages.

4.2.2. Joining of a New Peer

The joining of a new peer into the running MMVE takes place after the MMVE software is started on a user computer. Before the user is able to log in, all layers and services of the MMVE software including the architectural components for continuous event support are started.

After CEC, CEE, CR, TS and CES were started, the CEC is provided by the MMVE software with the peer id of the superpeer of the zone in order to be able to address network messages directly to the superpeer for continuous event management. Then the CR checks if new continuous event function code is available from other CRs in the system. In case new function code is found, the CR downloads the code.

After all new code was downloaded, the user is able to log in. After the log in, according to the target system model the MMVE software uses the SPS to perform a new AoI subscription for this peer in dependency of the initial position of the user avatar. Based on this AoI subscription, the CEM of the superpeer of the zone is able to detect AoEs of existing continuous events that intersect the AoI subscription according to the algorithm for management of continuous events, which is described in Subsection 4.2.5 of this work. In case the CEM detects an intersection, it sends a copy of the corresponding continuous event to the CEC of the newly joined regular peer and adds the peer id of the regular peer to the receiver list of the continuous event. The CEC of the newly joined peer then hands the received copy over to the CEE for execution. More details about the process of registering peers as receivers based on receiver lists and the processes of executing and managing continuous events are given later on in this work.

4.2.3. Leaving of a Peer

A peer is leaving the system intentionally in case a user logs out of the system and the MMVE software is shut down.

In case the peer is a regular peer, the CEC sends a message to the CEC of the superpeer of the zone. The CEC of the superpeer triggers the removal of the peer id of the regular peer from all receiver lists by the CEM. All architectural components for support of continuous events (CEC, CEE, CR, TS and CES) are shut down on the leaving regular peer. According to the previously discussed design of the CES no continuous event states are stored persistently. The next time the MMVE software

is started, the peer will be provided with up-to-date copies of all existing continuous events as described in Subsection 4.2.2.

In case the peer is a superpeer, according to the target system model the MMVE system selects another peer as superpeer for the zone before the MMVE software of the peer is shut down. In order to manage continuous events on the new superpeer, the MMVE software starts a CEM. Analogical to the system behavior that was described for MMVE start (see Subsection 4.2.1), the CEC of the new superpeer performs a spatial subscription for continuous event messages that corresponds to the shape of the zone. The CEC of the leaving superpeer generates a message that contains an aggregation of all managed continuous events and their receiver lists. Then the DL is used to provide the aggregation to the new superpeer. The CEC of the new superpeer receives the aggregation and hands it over to the CEM. The CEM starts managing the continuous events for the zone based on the provided informations. The CECs of all peers within the zone are provided with the peer id of the new superpeer for direct addressing of future network messages. After that, the CEC of the leaving superpeer revokes its spatial subscription for the continuous event messages of the zone. All architectural components for support of continuous events are shut down on the leaving superpeer. Analogical to a regular peer, no continuous event states or receiver lists are stored persistently.

4.2.4. Start and Execution of Continuous Events

In case a continuous action takes place on a peer, a new continuous event can be started explicitly by the MMVE software via the CEC. Figure 4.2 on page 78 gives the pseudocode for the starting of a new continuous event by the CEC. The CEC has to be provided with the following informations: The starting time of the continuous event, the interval length between the recalculations, the maximum number of intervals, the ids of the code for the functions, and optional parameter values for the functions for calculating the AoE and the effect over time. The CEC first creates a unique id and assigns it to the new continuous event (lines 2 and 3). Then the point in time of the next execution and management step is set to the given starting time (line 4). After that, the point in time of the last execution and management step is calculated and set based on the given informations if the maximum number of intervals is not infinite, or set to infinite else (lines 5 to 10). In the following, the other attributes of the new continuous event are filled with the given informations (line 11). At next, the code for calculating the AoE of the

```
   Input: starting time (t_0), interval length (δt), maximum number of intervals (n), id of
          f_AoE (aoeid), optional parameters of f_AoE (aoepms), id of f_E (eid), optional
          parameters of f_E (epms), id of f_apply (aid)
 1 begin
 2 |  ce ← initialize;
 3 |  id[ce] ← create unique id;
 4 |  nextstep[ce] ← t_0;
 5 |  if not n is infinite then
 6 |  |  finalstep[ce] ← t_0 + (n * δt);
 7 |  end
 8 |  else
 9 |  |  finalstep[ce] ← infinite;
10 |  end
11 |  set values of other attributes of ce: δt, aoeid, aoepms, eid, epms, aid;
12 |  aoefunction ← retrieve code for aoeid from CR;
13 |  initialaoe ← call aoefunction(t_0, aoepms);
14 |  use SPS to propagate ce to initialaoe;
15 |  hand ce over to CEE and start execution;
16 end
```

Figure 4.2.: Start of a new continuous event

continuous event is retrieved from the CR and the initial AoE is calculated based on the given informations (lines 12 and 13). The SPS is used to propagate the continuous event via a spatial propagation to the calculated AoE (line 14). After the spatial propagation was made, the continuous event is handed over to the local CEE for execution (line 15).

The SPS delivers the continuous event to all peers with a spatial subscription for continuous event messages that intersects the AoE. This includes all regular peers whose AoI subscription intersects the AoE, and always the superpeer because the CEC of the superpeer made an additional spatial subscription corresponding to the shape of the zone after the CEM was started. Please note that according to the given target system model the user of the superpeer also participates in the MMVE. Therefore, it is possible that the AoI subscription of this peer intersects with the AoE of a continuous event in addition to the zone-wide spatial subscription for continuous event management. As a result, this peer receives the continuous event twice. This system behavior is intended because execution and management of continuous events have to be clearly separated in order to be able to hand over the managed continuous events in case the superpeer changes. The system behavior for continuous event management that takes place at the superpeer is described later on in Subsection 4.2.5. The remainder of this subsection focuses on the reception and execution of continuous events by regular peers.

```
   Input: continuous event (ce)
1  begin
2      retrieve function code for aoeid[ce], eid[ce] and aid[ce] from CR;
3      while nextstep[ce] <= current MMVE time and nextstep[ce] <= finalstep[ce] do
4          aoe ← call aoefunction(nextstep[ce], aoepms[ce]);
5          e ← call efunction(nextstep[ce], epms[ce]);
6          singleevents ← call applyfunction(aoe, e);
7          hand singleevents over to MMVE software;
8          nextstep[ce] ← nextstep[ce] + δt[ce];
9      end
10     if nextstep[ce] < finalstep[ce] then
11         store ce in CES;
12         use TS to register new timer for id[ce] at nextstep[ce];
13     end
14     else
15         discard ce;
16     end
17 end
```

Figure 4.3.: Start of continuous event execution

After the CEC of a regular peer received a continuous event for execution from the SPS, it hands the continuous event over to the CEE. Then the CEC uses the DL to connect to the superpeer in order to register the peer as a receiver of the continuous event.

Figure 4.3 gives the pseudocode for the start of execution of a continuous event by the CEE after it was handed over from the CEC. At first, the function code is retrieved from the CR according to the ids included in the continuous event (line 2). After that, the CEE has to check if any execution steps were missed. Execution steps can be missed, for example, in case there is a high network latency between the propagating peer and the receiving peer and the propagation of the continuous event over the network takes longer than the given interval between execution steps. Therefore, the algorithm for starting the execution of a continuous event includes a mechanism that checks for missed steps and converges continuous events towards the current MMVE time. The mechanism can be described by using a while loop (lines 3 to 9). As long as the next planned execution step is before or simultaneous to the current MMVE time and before or simultaneous to the final execution step, the CEE performs the function calls, creates artificial single events, and hands them over to the MMVE software for further processing. After that, the planned next execution step is advanced according to the given interval. After the while loop was left, the algorithm checks if the while loop was only left because the current MMVE

```
   Input: id of continuous event (ceid)
1  begin
2      ce ← retrieve continuous event for ceid from CES;
3      retrieve function code for aoeid[ce], eid[ce] and aid[ce] from CR;
4      while nextstep[ce] <= current MMVE time and nextstep[ce] <= finalstep[ce] do
5          aoe ← call aoefunction(nextstep[ce], aoepms[ce]);
6          e ← call efunction(nextstep[ce], epms[ce]);
7          singleevents ← call applyfunction(aoe, e);
8          hand singleevents over to MMVE software;
9          nextstep[ce] ← nextstep[ce] + δt[ce];
10     end
11     if nextstep[ce] < finalstep[ce] then
12         update state of ce in CES;
13         use TS to register new timer for id[ce] at nextstep[ce];
14     end
15     else
16         remove ce from CES;
17     end
18 end
```

Figure 4.4.: Step of continuous event execution

time was reached and if there are still any execution steps left (line 10). In case there are any steps left, the continuous event is stored in the CES and a timer that triggers at the planned point in time of the next execution step is registered for the continuous event (lines 11 and 12). In case no more steps are left, the continuous event is discarded (line 15).

The pseudocode that describes a step of the following continuous event execution process can be derived straightforward from the described pseudocode for start of continuous event execution (see Figure 4.4). After the timer for the continuous event triggered, the previously stored state of the continuous event is retrieved from the CES based on the continuous event id that was attached to the timer (line 2). The function code is retrieved (line 3) and, analogical to the previously described algorithm, the CEE checks for missed execution steps and converges the continuous event towards the current MMVE time (lines 4 to 10). Execution steps can be missed during the regular execution, for example, if the resources of a peer get overloaded and the timer service that is used by an implementation of the system architecture cannot trigger its timers at the scheduled points in time. After the while loop was left, the CEE checks if there are any execution steps left (line 11). In case there are any steps left, the stored state of the continuous event is updated and a new timer is registered (lines 12 and 13). In case no more steps are left, the continuous event has reached the end of its lifetime and has to be terminated automatically. The

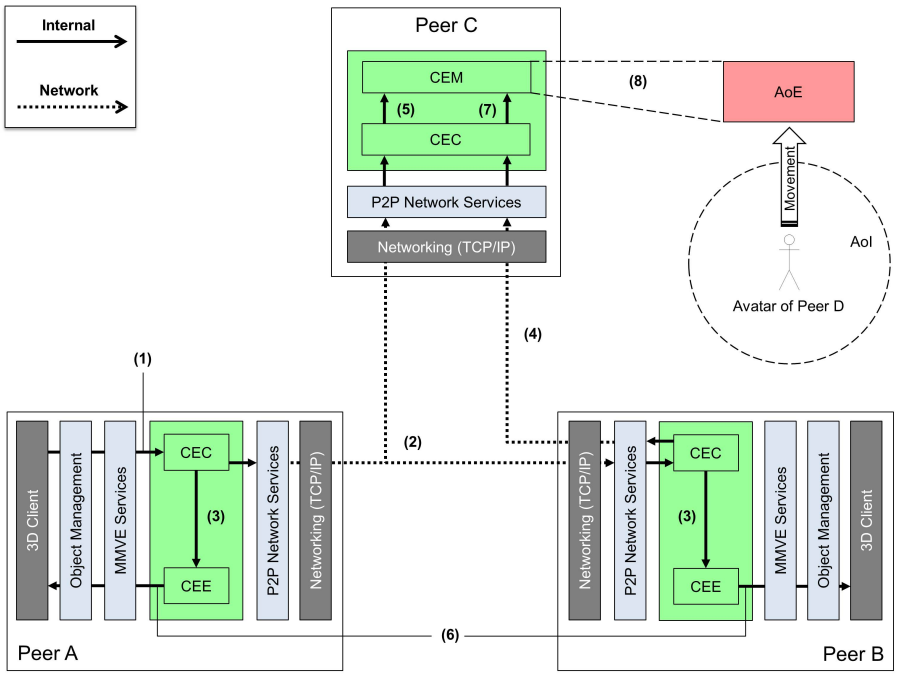


Figure 4.5.: Information flow for start and execution of continuous events

previously stored state is removed from the CES and no timer is registered (line 16).

Figure 4.5 presents a concluding overview of the system behavior and information flow in case a new continuous event is started by a CEC and, in the folllowing, executed by the CEEs of receiving peers and managed by the CEM of the superpeer of a zone. The avatars of peer A and peer B both are located in the zone of superpeer C. Assuming a continuous action takes place on peer A, a continuous event can be started explicitly via the CEC on peer A (1). As described by the algorithm presented in Figure 4.2, the CEC of peer A uses the given informations to create a continuous event. Then the continuous event is propagated to its AoE by using the SPS (2). The CECs of all peers with intersecting AoI subscriptions as well as the CEC of the superpeer, which made an additional spatial subscription for the zone, receive the continuous event. The CECs of all peers that received the continuous event (including peer B) hand the continuous event over to their CEEs for execution (3) and use the DL to send a direct network message to the superpeer C including their peer id and the continuous event's id in order to register as a receiver of this continuous event (4). The CEC of the superpeer hands the continuous event over to the CEM for management (5). On the regular peers, the CEEs perform the algorithm for handling missed function calls and the start of continuous event execution as described in Figure 4.3. After that, the standard execution process as

described in Figure 4.4 is started. In the following, the continuous event is executed and artificially created single events are handed over to the MMVE software for further processing (6). On the superpeer, after receiving the continuous event for management from the CEC, the CEM starts the management of the continuous event, creates a receiver list and adds the peer id of the peer that propagated the continuous event to the receiver list. After a message for registering as a receiver of a continuous event was received from the CEC of a regular peer, the CEC of the superpeer hands the information over to the CEM (7) and the CEM adds the peer id to the receiver list of the continuous event. In the following, the CEM checks for additional intersections of the current AoE of the continuous event with AoI subscriptions of peers in the zone (8). If the avatar of peer D moves closer to the AoE of the continuous event, for example, peer D is provided with an up-to-date copy. More about the system behavior and algorithms for continuous event management is given in the following subsection.

4.2.5. Management of Existing Continuous Events

Figure 4.6 on page 83 presents the pseudocode of the algorithm that is performed after a continuous event was received for management by the CEC of a superpeer and handed over to the CEM. The CEM first retrieves the function code for calculating the AoE and effect of the continuous event from the CR (line 2). Then missed management steps are detected and the continuous event is converged towards the current MMVE time, similar to the algorithm for starting the execution of a continuous event. Using a while loop, the functions are called and the scheduled point in time of the next management step is advanced until either the current MMVE time or the point in time of the last management step is reached (lines 3 to 7). In contrast to the algorithm for starting the execution, no artificial single events are created and handed over to the MMVE software. After the while loop was left, the CEM checks if there are any management steps left (line 8). In case there are any steps left, a new empty receiver list is created for the continuous event, the peer id of the propagating peer is added to the list, and the list is added the the collection of receiver lists of the CEM (lines 9 to 11). The continuous event is stored in the CES (line 12) and a timer that triggers at the planned point in time of the next management step is registered for the continuous event (line 13). In case there are no management steps left, no receiver list has to be created and no timer has to be registered. The continuous event is discarded (line 16).

```
   Input: continuous event (ce)
1  begin
2      retrieve function code for aoeid[ce] and eid[ce] from CR;
3      while nextstep[ce] <= current MMVE time and nextstep[ce] <= finalstep[ce] do
4          call aoefunction(nextstep[ce], aoepms[ce]);
5          call efunction(nextstep[ce], epms[ce]);
6          nextstep[ce] ← nextstep[ce] + δt[ce];
7      end
8      if nextstep[ce] < finalstep[ce] then
9          rlist ← create empty receiver list for id[ce];
10         add peer id of propagating peer to rlist;
11         add rlist to existing receiver lists;
12         store ce in CES;
13         use TS to register new timer for id[ce] at nextstep[ce];
14     end
15     else
16         discard ce;
17     end
18 end
```

Figure 4.6.: Start of continuous event management

As mentioned earlier, superpeers additionally have to perform the tasks of a regular peer in order to allow their users to participate in the MMVE. In order to separate the execution and management clearly, superpeers can receive continuous events twice if the AoI subscription of the superpeer and the additional spatial subscription by the CEC for continuous event messages of the zone both intersect the AoE of a continuous event. This has to be considered in the context of storing continuous events because based on the presented algorithms both versions of the continuous event, for execution and management, have the same continuous event id. In order to store the continuous event twice in the CES and to identify the copy for execution and the copy for management, a modification is needed. Such a modification is not included in the presented algorithms for execution and management, but certainly has to be considered when implementing the presented system architecture. For example, the software prototype that was used for evaluating the continuous events approach (see Chapter 5) stores continuous events in a hash table und uses the continuous event id as key. In order to divide between the continuous event copy for execution and management, the CEM adds a prefix to the continuous event id before a continuous event is stored or retrieved.

When managing continuous events, crucial tasks are the detection of new intersections between the AoEs of managed continuous events and AoI subscriptions of peers and the provision of up-to-date copies of continuous events to peers with new

intersections. As mentioned earlier in the section about the overall design of the system architecture, the CEM is able to use the available AoI subscriptions from the SPS of the target system because it is placed on the existing superpeer of the target system. This eliminates the need for any additional network messages. Concerning the design of an algorithm for continuous event management, two alternatives come to mind. Either the CEM is automatically provided with all new or updated AoI subscriptions after they are received by the SPS, checks immediately for intersections with the AoEs of managed continuous events and, in case a new intersection is found, provides the peers that have an intersecting AoI and are not included in the receiver list directly with the state of the continuous event that is stored in the CES. Or the CEM retrieves the currently available AoI subscriptions from the SPS after the timer for management of a continuous event triggered and the AoEs were recalculated, checks for intersections between the AoI subscriptions and the AoEs and provides all peers that have an intersecting AoI and are not included in the receiver list of the continuous event with a copy of the continuous event.

The former alternative only checks for intersections if there are new or updated incoming AoI subscriptions. This potentially optimizes the number of intersection checks because no checks are performed between existing continuous events and unchanged AoIs. If the peers do not move, no updated AoI subscriptions are sent to the SPS and no check is performed by the CEM. However, this alternative leads to problems in case the AoE of a continuous event changes. If the AoE of a continuous event changes their location or shape, this can result in new intersections with existing unchanged AoI subscriptions. This is not detected if a check for intersections between the AoEs of managed continuous events and AoI subscriptions is only performed when the SPS receives new or updated AoI subscriptions.

The latter alternative checks for intersections whenever the functions are called during the management process and the continuous event changes its state. The check for intersections detects new intersections that were caused by changing AoEs and by new or updated AoI subscriptions because the current state of the AoE of a continuous event was just calculated and a retrieval of available AoI subscriptions at that time returns a current state of all AoIs. On the downside, this alternative potentially results in a larger number of intersection checks and calculation operations in comparison to the former alternative because intersection checks between the AoE of a continuous event and all available AoI subscriptions have to be performed whenever the timer of the continuous event triggers.

```
   Input: id of continuous event (ceid)
 1 begin
 2     ce ← retrieve continuous event for ceid from CES;
 3     rlist ← get receiver list for id[ce] from existing receiver lists;
 4     retrieve function code for aoeid[ce] and eid[ce] from CR;
 5     aoe ← initialize;
 6     while nextstep[ce] <= current MMVE time and nextstep[ce] <= finalstep[ce] do
 7         aoe ← call aoefunction(nextstep[ce], aoepms[ce]);
 8         call efunction(nextstep[ce], epms[ce]);
 9         nextstep[ce] ← nextstep[ce] + δt[ce];
10     end
11     aois ← retrieve aoi subscriptions from SPS;
12     foreach aoi in aois do
13         if aoe intersects shape[aoi] then
14             if not rlist contains peerid[aoi] then
15                 use DL to provide peerid[aoi] with ce;
16                 add peerid[aoi] to rlist;
17             end
18         end
19     end
20     if nextstep[ce] < finalstep[ce] then
21         update state of ce in CES;
22         use TS to register new timer for id[ce] at nextstep[ce];
23     end
24     else
25         remove rlist from existing receiver lists;
26         remove ce from CES;
27     end
28 end
```

Figure 4.7.: Step of continuous event management

In conclusion, only the latter alternative makes sure that intersections between changing AoEs and non-changing AoIs are detected. This alternative has to be used by the CEM, although it results in a potentially higher calculation load.

Figure 4.7 gives the pseudocode for a step of the continuous event management process. The algorithm is performed after the timer for a continuous event triggered. At first, the continuous event state, the receiver list and the function code are retrieved (lines 2 to 4). Then missed management steps are detected and the continuous event is converged towards the current MMVE time (lines 6 to 10), analogical to the algorithm for starting the management of continuous events. After that, new intersections between the AoE of the continuous event and available AoI subscriptions are detected and copies of the continuous event are provided (lines 11 to 19). The available AoI subscriptions are retrieved from the SPS. For each AoI, a check is performed if the AoE of the continuous event intersects the AoI. In case

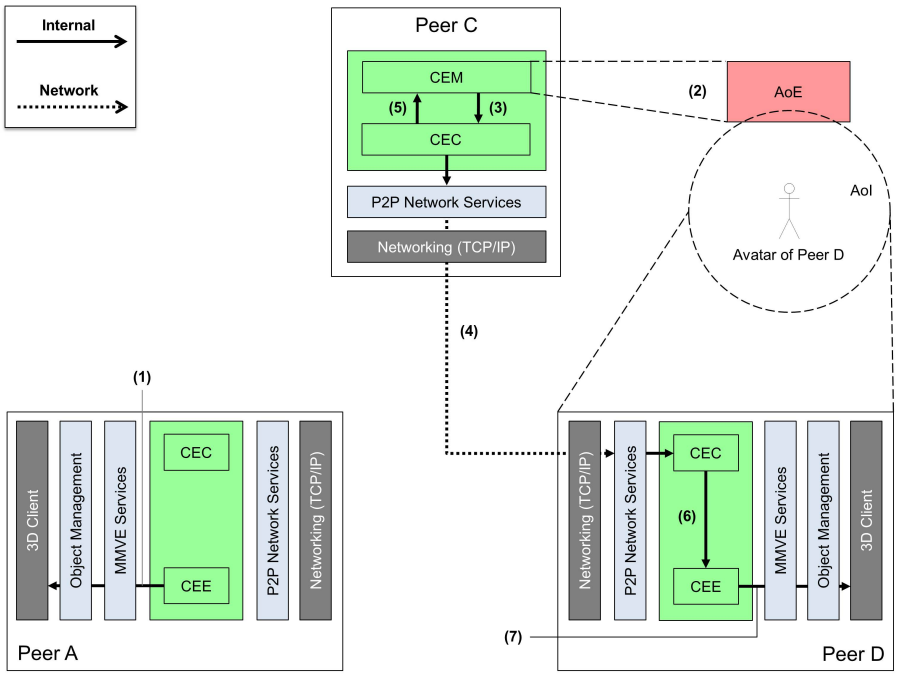


Figure 4.8.: Information flow after detection of a new intersection between AoE and AoI

an intersection is detected, the algorithm checks if the receiver list contains the id of the corresponding peer. If the list does not contain the peer id, the DL is used to provide an up-to-date copy of the continuous event to the peer and the peer id is added to the receiver list. After the detection of new intersections was performed, the algorithm determines if there are any future management steps left (line 20). In case there are any steps left, the state of the continuous event is updated and a new timer is registered (lines 21 and 22). In case there are no steps left, the continuous event has to be terminated automatically. The receiver list is removed from the collection of existing receiver lists of the CEM, no new timer is registered, and the previously stored state of the continuous event is removed from the CES (lines 25 and 26).

Figure 4.8 illustrates the information flow in the MMVE system in case an intersection between the AoE of an existing continuous event and an AoI subscription is detected and the corresponding peer is not contained in the receiver list of the continuous event. The figure assumes that the continuous event whose creation and start of execution was shown earlier in Figure 4.5 is still executed and managed. For example, peer A is one of the peers that execute the continuous event (1). Superpeer C manages the continuous event. Since the creation and start of execution and management of the continuous event, the avatar of peer D has moved closer to the AoE

of the continuous event and has updated its AoI subscription at superpeer C. As a result, the corresponding AoI subscription now intersects the AoE. As part of the managing process, the CEM of superpeer C checks regularly for new intersections between AoEs and AoIs according to the algorithm described by Figure 4.7. After the timer for the continuous event triggered, the intersection is detected based on the given algorithm (2). The CEM hands an up-to-date copy of the continuous event and the id of peer D over to the CEC (3). The CEC handles the communication. It uses the DL to provide the CEC of peer D with the copy (4). In case the sending was successful, the CEC of superpeer C informs the CEM and the id of peer D is added to the receiver list of the continuous event (5). The CEC of peer D hands the received continuous event copy over to the CEE (6) and, in the following, the CEE of peer D executes it (7). Because the id of peer D was added to the receiver list, the CEM does not provide peer D with another copy of the continuous event anymore, even the AoI still intersects the AoE.

4.2.6. Modification and Termination of Existing Continuous Events

Informations about the modification or termination of a continuous event have to be sent to all peers that execute a copy because their copies have to be adjusted. In addition, the copy that is used for management by the CEM has to be adjusted as well. The information flow in the system for sending informations about the modification or termination of a continuous event is identical. Therefore, both cases are described together in this subsection.

Existing continuous events can be modified or terminated on one of the executing peers. This can happen, for example, because a user input interrupts a continuous action. Another example is that a peer controls weather effects in an area, decides to change wind direction and has to adjust the movement direction of clouds that are represented as continuous events in the system. In the system architecture for continuous event support, existing continuous events can be modified or terminated explicitly via the CEC. In case of continuous event modifications, the id of the continuous event as well as the modifications have to be provided to the CEC. By design, the system architecture for continuous event support allows the modification of specific attributes of continuous events. The CEC has to be provided with the continuous event id and a collection of pairs that describe the modifications of specific attributes. Each pair includes the name of the attribute and the new value. The CEC sends all provided modifications in one modification message to the CEC

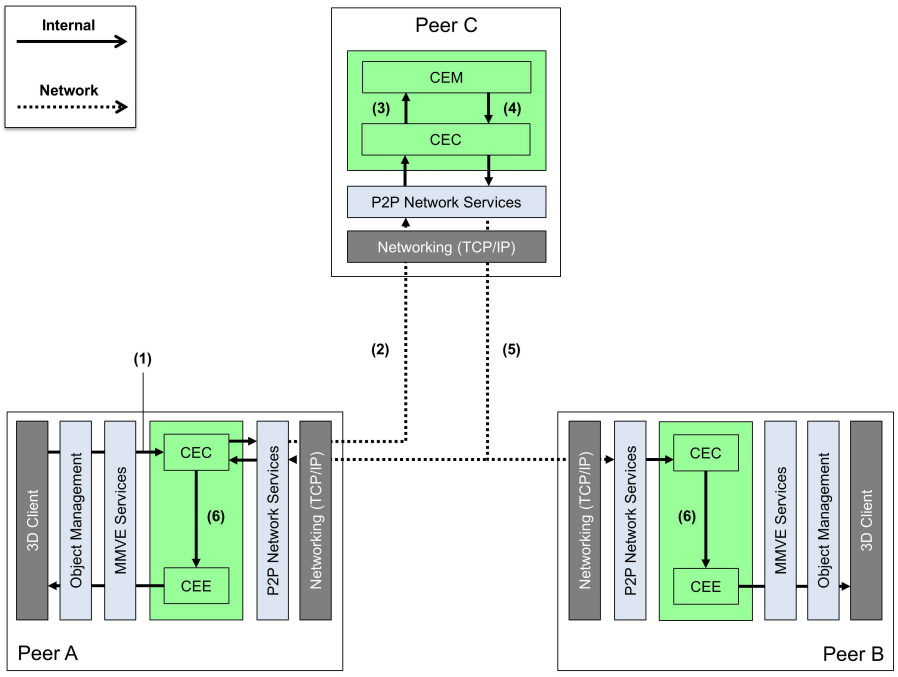


Figure 4.9.: Information flow for modification or termination of existing continuous events

of the superpeer. The CEC of the superpeer then propagates the message based on the receiver list of the continuous event to the CECs of the peers that hold a continuous event copy. In case of a continuous event termination, the CEC only has to be provided with the continuous event id. The CEC sends a termination message including the id to the CEC of the superpeer. The CEC of the superpeer then propagates the message based on the receiver list of the continuous event to the CECs of the peers that hold a continuous event copy.

The resulting information flow in the MMVE system is identical for modifications and terminations. Figure 4.9 gives an overview of the information flow. The figure assumes that peer A and peer B both execute the continuous event. Superpeer C manages the continuous event. After a continuous event was explicitly modified or terminated via the CEC of peer A (1), the CEC of peer A creates the corresponding message type and uses the DL to provide the message directly to the CEC of superpeer C (2). The CEC of superpeer C provides the modification or termination information to the CEM. The CEM modifies or terminates its copy of the continuous event (3). In addition, the CEC retrieves the receiver list of the continuous event from the CEM (4) and uses the DL to provide the CECs of all peers that are included in the list (including peer A) with the modification or termination message (5). The CECs of the peers then provide the CEEs with the modification or termi-

nation information. The CEEs modify or terminate their copies of the continuous event (6).

From a theoretical point of view, there is another alternative to the described information flow. The CEC of peer A can provide the CEE of peer A directly with the modification or termination message after the sending of the message to the CEC of superpeer C was successful. In comparison to the previously described information flow, this alternative results in one less network message because peer A does not need to be provided with the modification or termination message by the CEC of superpeer C. However, this alternative results in a deviation between peer A and other peers that execute the continuous event, for example peer B, because of the additional hop via the CEC of superpeer C. In order to prevent this, the potential overhead of the additional message is accepted at the benefit of eliminating state deviations between the peers caused by the continuous events approach. Please note that there can still be slight deviations between the peers because of varying latencies of the connections between the executing peers and the superpeer. However, in the context of the target system varying latencies of the connections between the regular peers and the superpeer of a zone also affect standard single events. The handling of latency in the overall context of the target system is beyond the scope of this work.

4.3. Extensions

The previous section described the system behavior and algorithms for basic support of continuous events by the system architecture. The focus of this work is on the general concept of continuous events in P2P-MMVEs and its potential usage for a scalable propagation of continuous actions. The previously described basic continuous event support by the system architecture provides all functionalities that are needed for using the concept of continuous events in P2P-MMVEs and for propagating continuous actions with continuous events. The evaluation of the potential of the concept of continuous events and its usage in P2P-MMVEs, which is presented later on in Chapter 5, was performed based on the system architecture for basic continuous event support. Nevertheless, the existence of multiple zones in the target system model and the handling of certain exceptional cases that are included in the requirements for this work imply a need to discuss several extensions for basic continuous event support. These extensions are discussed in the remainder of this

section. However, they are not elaborated in the same detail as basic continuous event support. The following extensions are presented and discussed: Extensions for support of multiple zones (Subsection 4.3.1), the handling of peer crashes (Subsection 4.3.2), the handling of peer disconnections (Subsection 4.3.3) and the handling of overloaded peers (Subsection 4.3.4).

4.3.1. Extensions for Support of Multiple Zones

In the context of a P2P-MMVE with multiple zones, for example the target system described in Section 2.3, several exceptional cases can occur that are mainly related to the existence of zone borders. In the following, the implications of multiple zones on the basic algorithms for start, execution, management, modification and termination of continuous events are discussed and potential extensions for the basic algorithms are presented.

Start and Execution of Continuous Events

In a P2P-MMVE with multiple zones, the initial AoEs of newly started continuous events can be located in several zones. Figure 4.10 on page 91 gives an overview of the potential cases that can occur. The figure assumes that the P2P-MMVE consists of four zones. Avatar A is located in zone 1. A continuous action is performed by avatar A and is represented by a continuous event. The initial AoE of the continuous event that is started on the corresponding peer of avatar A can be located fully in the same zone (1), partially in the same zone and partially in one other zone (2), partially in the same zone and partially in multiple other zones (3), fully in another zone (4), or fully outside of the same zone and partially in multiple other zones (5).

In the context of basic continuous event support, the initial AoE is used for a spatial propagation of the continuous event. The spatial propagation is made by using the SPS of the existing target system. The spatial publication makes sure that all regular peers whose AoI subscription intersects the AoE and the superpeer of the zone, which made an additional spatial subscription for continuous event messages of the zone, receive the continuous event for further execution and management.

The existing SPS delivers a spatial publication to all peers with spatial subscriptions that intersect the given spatial shape. Based on this, all cases can be supported by basic continuous event support without any further extensions. All regular peers

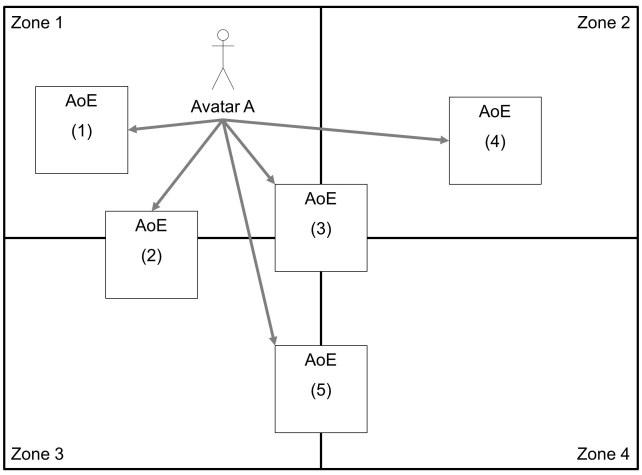


Figure 4.10.: Potential AoE locations for newly started continuous events in P2P-MMVEs with multiple zones

with intersecting AoI subscriptions receive the spatial publication from the SPS, no matter if the peers are located in the same zone or in another zone. In addition, assuming the CECs of the superpeers of the four zones all performed spatial subscriptions for continuous event messages of their zones as described in Section 4.2.1 about the MMVE start, they also receive the spatial publication from the SPS in case the AoE intersects with their zone. In conclusion, no extensions have to be made concerning the start and initial propagation of continuous events.

Concerning the following execution and management of continuous events, a propagation of a continuous event to regular peers and superpeers of other zones results in continuous events that are executed by the CEEs of peers in multiple zones and managed simultaneously by the CEMs of superpeers of multiple zones. The simultaneous management of a continuous event by multiple CEMs creates a need for coordination between the CEMs. This is discussed later on as part of the discussion about implications for continuous event management. In the following, this work focuses on the execution and the receiver registrations that are performed as part of the execution.

According to the described basic continuous event support, the CEC of a regular peer registers as a receiver by sending a message to the CEC of the superpeer of its zone after it received a continuous event. This can also be supported in multiple zones without any extensions because the CEC of the superpeer of a zone receives continuous events that are created in other zones from the SPS in case the AoE

```
   Input: starting time (t_0), interval length (δt), maximum number of intervals (n), id of
          f_AoE (aoeid), optional parameters of f_AoE (aoepms), id of f_E (eid), optional
          parameters of f_E (epms), id of f_apply (aid)
 1 begin
 2 |  ce ← initialize;
 3 |  id[ce] ← create unique id;
 4 |  nextstep[ce] ← t_0;
 5 |  if not n is infinite then
 6 |  |  finalstep[ce] ← t_0 + (n * δt);
 7 |  end
 8 |  else
 9 |  |  finalstep[ce] ← infinite;
10 |  end
11 |  set values of other attributes of ce: δt, aoeid, aoepms, eid, epms, aid;
12 |  aoefunction ← retrieve code for aoeid from CR;
13 |  initialaoe ← call aoefunction(t_0, aoepms);
14 |  use SPS to propagate ce to initialaoe;
15 |  zone ← get current zone from MMVE software;
16 |  if not initialaoe intersects shape[zone] then
17 |  |  use DL to provide superpeerid[zone] with ce;
18 |  end
19 |  hand ce over to CEE and start execution;
20 end
```

Figure 4.11.: Extended start of a new continuous event

intersects its zone-wide spatial subscription. Therefore, the CEM of the superpeer is able to manage such continuous events and peers in the zone can register as receivers.

A problem can occur if the AoE for initial propagation is located fully outside of the zone of the peer that created the continuous event. In this case, the CEC of the superpeer of the original zone is not provided with the continuous event via the zone-wide spatial subscription and the CEM of this peer does not manage the continuous event. This is a problem because the peer that created the continuous event has to execute it to allow further modifications and termination on this peer. Because the peer executes the continuous event, according to the described basic continuous event support it has to register with the CEM of its zone as a receiver. Because the CEM of the superpeer of its zone does not manage the continuous event, the peer can not register as a receiver. An extension for the basic algorithm for starting new continuous events is needed.

Figure 4.11 gives the pseudocode for an extended algorithm. After the CEC performed the spatial publication (line 14), it retrieves the shape of the current zone of the peer from the MMVE software (line 15). It checks if the AoE intersects the

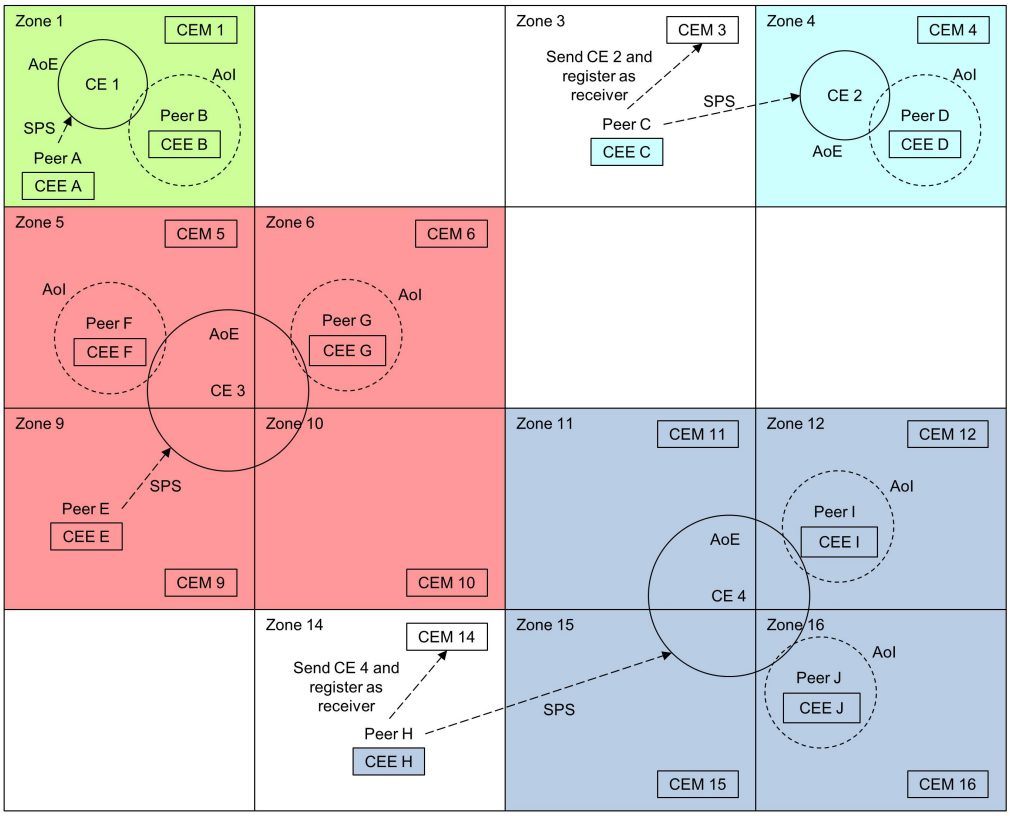


Figure 4.12.: Continuous event propagation in P2P-MMVEs with multiple zones

shape of the zone. If the AoE does not intersect the shape of the zone at all, the CEC of the superpeer of the zone did not receive a continuous event copy via the spatial propagation. Therefore, the CEC of the superpeer is provided with a copy via the DL (lines 16 to 18).

In addition to the given explanation and extended algorithm, Figure 4.12 gives a detailed illustration of the potential locations of AoEs and propagation of new continuous events in P2P-MMVEs with multiple zones. Figure 4.12 shows a P2P-MMVE with 16 zones. The peers A, C, E and H create new continuous events.

Peer A creates a new continuous event CE 1 that is propagated to a circular AoE. The AoE of CE 1 intersects the AoI of peer B and is fully located within the same zone where the avatar of peer A is located (zone 1, green). The propagation to the AoE according to basic continuous event support is able to provide a copy of CE 1 to all CEEs (CEE B) and all CEMs (CEM 1) that need a copy. No extension to basic continuous event support is needed.

Peer E creates a new continuous event CE 3 that is propagated to a circular AoE.

The AoE of CE 3 intersects the AoIs of peer F and G as well as the zones 5, 6, 9 and 10 (red). Again, no extension to basic continuous event support is needed. The propagation to the AoE provides a copy of CE 3 to all CEEs (CEE F and G) and CEMs (CEM 5, 6, 9 and 10) that need a copy.

An extension for basic continuous event support is needed to support continuous events with AoEs that are located fully outside of the original zone. Peer C creates a new continuous event CE 2 that is propagated to a circular AoE. The AoE intersects the AoI of peer D and zone 4. It is located fully outside of the original zone (zone 3). The basic continuous event support algorithm provides the CEE of peer D and the CEM of zone 4 with a copy of CE 2 via the spatial publication to the AoE (light blue). In addition, it starts the execution by the CEE of peer C (light blue). However, peer C is not able to register as receiver of modification and termination messages because the CEM of its zone is not provided with a copy of CE 2 via the spatial publication. Therefore, basic continuous event support has to be extended and peer C has to provide CEM 3 directly with a copy of CE 2.

The same extension is able to support continuous events with AoEs that are located fully outside of the original zone and intersect multiple other zones, for example CE 4 that is created and propagated by peer H. The spatial propagation to the AoE provides a copy of CE 4 to the CEEs of peer I and J as well as the CEMs of the zones 11, 12, 15 and 16 (dark blue). In addition, the execution of CE 4 by CEE H is started (dark blue). Because of the extension, peer H provides the CEM of zone 14 directly with a copy of CE 4 and peer H is able to register as a receiver of CE 4.

Management of Continuous Events

The AoEs of continuous events can change location and spatial shape over time depending on the results of the called function code. In a P2P-MMVE with multiple zones, this can result in AoEs that leave the zone of the CEM that initially managed the continuous event. This has implications for the management of continuous events because in the algorithms for basic continuous event support the task of managing a continuous event is assigned based on the intersection between the continuous event's AoE and the zone. In case the AoE of an existing continuous event leaves a zone over the lifetime of the continuous event, this has to be detected and a mechanism for handing the management of the continuous event over from the CEM of the original zone to the CEMs of other zones is needed. The previously presented algorithms

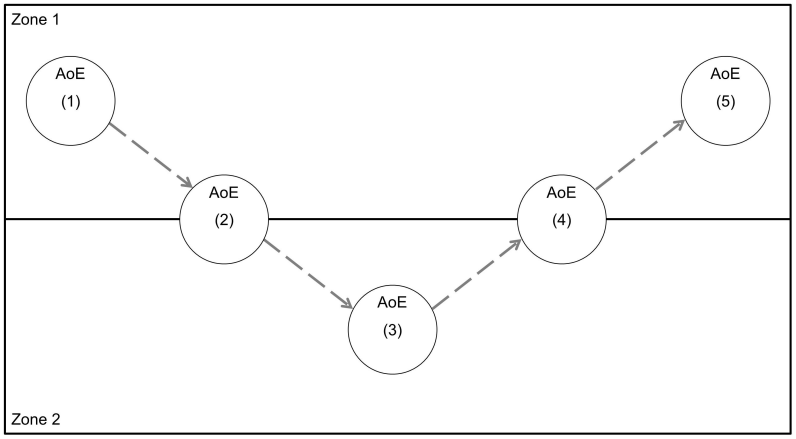


Figure 4.13.: Potential AoE locations for existing continuous events in P2P-MMVEs with multiple zones

for basic continuous event support are not able to handle this case without any extensions.

Figure 4.13 gives an overview of potential cases of AoEs that change zones. The figure assumes that the P2P-MMVE consists of two zones. A continuous action was performed on one of the peers of zone 1 and a corresponding continuous event was created. Because the initial AoE of the continuous event fully intersected zone 1 and did not intersect zone 2, only the CEM of the superpeer of zone 1 was provided with the continuous event and manages it. The CEM of the superpeer of zone 2 did not receive the continuous event. The following cases can be identified: The AoE stays within zone 1 for the whole lifetime of the continuous event (1), the AoE leaves the zone partially over time (2), the AoE leaves the zone fully over time (3), the AoE leaves and returns partially to the initial zone over time (4), or the AoE leaves and returns fully to the initial zone over time (5).

In case the AoE of a continuous event stays within a single zone for the whole lifetime of the continuous event (1), no extensions are needed and the continuous event can be managed based on the previously described algorithms for basic continuous event support.

The basic algorithms are also able to manage continuous events whose AoEs have left the original zone and are located fully within another zone (3) or have left the original zone and returned fully to the zone (5). Extensions to the algorithms for basic continuous event support are needed to handle the change of zones (2) (4).

To be able to handle the change of zones, a CEM has to determine if the AoE of a

managed continuous event leaves the zone that is managed by the CEM. The target zone and its CEM have to be identified and this CEM has to be provided with a copy of the continuous event for future management. From a theoretical point of view, there are two alternative procedures concerning the CEM of the old zone after the CEM of the new zone was provided with a continuous event copy and the AoE has fully left the original zone. Either the CEM stops management of the continuous event or it resumes management.

Stopping the management results in less calculations and disburdens the hardware of the superpeer. On the downside, stopping management creates the need of another handover if the AoE alternates between zones and returns to the original zone in the future. In case of a continuous event that repeatedly alternates between zones, the CEMs have to exchange continuous event copies multiple times over the network. Stopping the calculation can also lead to a problem in case there are peers in the original zone that still execute the continuous event after a handover between the CEMs. If management is stopped after the handover, the CEM of the original peer is not able to handle communication if the continuous event gets modified or terminated on one of these peers.

Resuming management makes sure that all CEMs that once received a continuous event have an up-to-date state. The resulting calculation overhead for continuous events that have left zones puts a higher burden on the hardware of the superpeers in comparison to the alternative of stopping management. However, less continuous event copies have to be exchanged between CEMs over the network in case of alternating AoEs. In addition, the described problem resulting from executing peers in the initial zone does not occur. In conclusion, resuming management is the more beneficial alternative. The extended algorithm for continuous event management, which is presented in the following, resumes management of continuous events after a handover.

In order to be able to handle AoEs that change zones over time, the existing algorithm for continuous event management by the CEMs has to be extended with a mechanism that checks for leaving AoEs and provides other CEMs with a copy of the corresponding continuous event if a leaving AoE is detected. Other CEMs can only be provided with continuous event copies if the CEM of the original zone is able to determine into which zone an AoE leaves and which superpeer is assigned to the zone because these informations are needed for addressing the network messages. Because the tesselation and assignment of superpeers of the target system is reused

```
   Input: id of continuous event (ceid)
1  begin
2      ce ← retrieve continuous event for ceid from CES;
3      rlist ← get receiver list for id[ce] from existing receiver lists;
4      retrieve function code for aoeid[ce] and eid[ce] from CR;
5      aoe ← initialize;
6      while nextstep[ce] <= current MMVE time and nextstep[ce] <= finalstep[ce] do
7          aoe ← call aoefunction(nextstep[ce], aoepms[ce]);
8          call efunction(nextstep[ce], epms[ce]);
9          nextstep[ce] ← nextstep[ce] + δt[ce];
10     end
11     ownzone ← get current zone from MMVE software;
12     if aoe intersects shape[ownzone] then
13         aois ← retrieve aoi subscriptions from SPS;
14         foreach aoi in aois do
15             if aoe intersects shape[aoi] then
16                 if not rlist contains peerid[aoi] then
17                     use DL to provide peerid[aoi] with ce;
18                     add peerid[aoi] to rlist;
19                 end
20             end
21         end
22     end
23     if not aoe fully intersects shape[ownzone] then
24         zones ← get zones from MMVE software;
25         foreach zone z in zones do
26             if aoe intersects shape[z] and not rlist contains superpeerid[z] then
27                 use DL to provide superpeerid[z] with ce;
28                 add superpeerid[z] to rlist;
29             end
30         end
31     end
32     if nextstep[ce] < finalstep[ce] then
33         update state of ce in CES;
34         use TS to register new timer for id[ce] at nextstep[ce];
35     end
36     else
37         remove rlist from existing receiver lists;
38         remove ce from CES;
39     end
40 end
```

Figure 4.14.: Extended step of continuous event management

by the continuous events approach, the informations about zone shapes and the ids of superpeers that are assigned to these zones can be retrieved from the SPS of the target system.

Figure 4.14 gives the pseudocode of an extended version of the algorithm for management of continuous events by CEMs. After all informations for management of

a continuous event were retrieved and the continuous event was converged towards the current MMVE time in case of missed function calls (lines 2 to 10), the spatial shape of the zone that is managed by the CEM is retrieved from the MMVE software (line 11). Because the existence of multiple zones implies that the new AoE of the continuous event can be fully located outside of the zone, the extended version of the algorithm first checks if the new AoE intersects the zone at all (line 12). Only if there is an intersection, the AoI subscriptions are retrieved from the SPS and the check for new intersections between the AoE and the AoI subscriptions is performed (lines 13 to 21). If the AoE is located fully outside of the zone, there is no need to waste hardware resources of the superpeer for performing the check for intersections.

At next, the algorithm checks if the AoE is fully located in the zone of the CEM (line 23). In case it is not fully located in the zone, parts of the AoE or the whole AoE have to be located in other zones and the CEMs of these zones need to be provided with a copy of the continuous event. The mechanism that determines intersections with other zones and provides the CEMs of these zones with a continuous event copy is described in lines 24 to 29. At first, all available zones are retrieved from the SPS (line 24). After that, the algorithm checks for all zones if there is an intersection with the AoE. In case an intersection is detected and the receiver list does not contain the id of the superpeer of the zone, the DL is used to provide a copy to the CEM of this superpeer and the id of the superpeer is added to the receiver list (lines 25 to 30). Please note that the superpeer of the other zone that received the copy starts management of the continuous event, creates a receiver list for the continuous event and adds the id of the providing superpeer to the receiver list in order to be able to address modification and termination messages back to the providing superpeer. This is not included in the given algorithm that describes a step of the extended continuous event management by the CEM.

After potential intersections with other zones were determined, continuous event copies were provided and the receiver list was adjusted, a step of extended continuous event management ends similar to a step of basic continuous event management. If there are any management steps left, the continuous event state is stored in the CES and a new timer is registered (lines 32 to 35). If there are no steps left, the receiver list is removed, no new timer is registered and the previously stored state of the continuous event is removed from the CES (lines 36 to 39).

Modification and Termination of Continuous Events

In a P2P-MMVE with multiple zones, continuous events can exist simultaneously in several of these zones. Continuous events that exist in several zones have to be managed by multiple CEMs and executed by multiple CEEs. In case such a continuous event is modified or terminated explicitly on one of the peers, all CEMs and CEEs with a copy of the continuous event have to be informed about this.

According to the basic information flow for modification and termination of continuous events, the peer where the continuous event was modified or terminated first sends the information to the superpeer of its zone. Based on the receiver list, the superpeer then sends the information to all peers within the zone that execute the continuous event. In a P2P-MMVE with multiple zones, this information has to be provided additionally to superpeers of other zones that manage the continuous event. These superpeers then can distribute the information to the executing peers within their zones according to the basic information flow.

The basic information flow for modification and termination of continuous events in P2P-MMVEs with multiple zones is extended for the support of multiple zones as follows: After a continuous event was modified or terminated explicitly via the CEC of an executing peer, the CEC uses the DL to send the information to the CEC of the superpeer of its zone. The CEC either hands the information over to the CEM for application of the modifications to the copy of the continuous event or it triggers the stopping of management of the continuous event by the CEM. In addition, the CEC retrieves the receiver list and sends the information to all peers whose ids are contained in the list. Because the ids of superpeers of other zones that received a copy are also included in the list, the sending to all peers in the list makes sure that all peers in the zone that execute the continuous event and all superpeers of other zones that manage the continuous event are provided with the information about modifications or termination. On the executing peers, the CEC triggers the stopping of execution of the continuous event by the CEE. On the superpeers of other zones, the CEC proceeds as described for the superpeer of the zone where the continuous event was modified or terminated.

Because the extension to the basic information flow for modification and termination of continuous events, which was illustrated by Figure 4.9 on page 88, is straightforward, no additional figure that illustrates the extended information flow is given.

4.3.2. Handling of Peer Crashes

With regards to the target system model, peer crashes can take place either because the MMVE software crashes or the user computer crashes. In both cases, the peer leaves the P2P-MMVE unintended and does not return for a certain period of time. During a peer crash, no algorithms can be performed by the MMVE software. Generally, a peer crash can affect a regular peer or a superpeer.

In a P2P-MMVE system, the crash or disconnection of a peer typically affects the system as a whole. Therefore, a mechanism that detects missing peers has to be part of the overall P2P-MMVE system and not only of continuous event support. In the following, it is assumed that there is a mechanism that detects if a peer is missing and if the missing peer is disconnected or has crashed.

In the context of continuous event support, handling the *crash of a regular peer* can be performed as follows: After the crash of a regular peer was detected by the overall P2P-MMVE system, the CEMs get informed and remove the id of the crashed peer from the existing receiver lists. In case a peer crashes, the MMVE software has to be restarted explicitly by the user, no matter if the MMVE software or the user computer crashes. As was described earlier, no continuous event states are stored persistently by the CES. Therefore, the CES is empty after the user restarts the MMVE software. In addition, the TS is reset and no more timers are present. According to the standard procedure of the target system, the peer performs a new AoI subscription based on the position of the user avatar. After that, intersections between the AoI subscription and the AoEs of existing continuous events are detected by the presented algorithm for continuous event management. Because the peer id was removed from the existing receiver lists, the CEM provides the peer with up-to-date copies of the continuous events and adds the peer id to the receiver lists.

In the target system of this work, the superpeers perform several system-related tasks for their zone in addition to just managing continuous events. Therefore, handling the *crash of a superpeer* has to be performed by the overall P2P-MMVE system instead of the system part for continuous event support. A detailed elaboration of a concept and algorithms that handle superpeer crashes for the overall P2P-MMVE system is beyond the scope of this work. A rough outline of a concept based on a replication strategy can be described as follows: One or multiple copies of the superpeer of a zone, including the architectural components for continuous event

support, are held on other regular peers whose hardware and networking capabilities are strong enough to be potentially a superpeer. In case the original superpeer crashes, one of the copies is used instead. Assuming that the replication of the superpeer and the takeover by a backup after a superpeer crash is performed by the overall system, the system has to make sure that the CEC of the new superpeer provides the CECs of all regular peers of the zone with its peer id in order to enable the future addressing of messages from the CECs of the regular peers to the CEC of the new superpeer via the DL.

4.3.3. Handling of Peer Disconnections

Similar to a peer crash, a peer disconnection from the peer-to-peer network can result in a peer that leaves the system unintended for a certain period of time and can affect a regular peer or a superpeer. In contrast to a peer crash, a peer disconnection can result in a state in which the MMVE software is still running and only lost connection to the network. If this state persists for a very short period of time, the simulation of the virtual environment by the MMVE software can continue. For example, the avatar and the virtual environment can still be displayed to the user. If the disconnection persists for a longer period of time, the simulation has to be stopped explicitly by the MMVE software because the displayed state varies strongly from the other peers. Both cases, short disconnections and long disconnections, affect the continuous events approach. Potential concepts for handling disconnections in the context of the continuous events approach are discussed in the following. Analogical to the discussion about the handling of peer crashes, the following discussion assumes that the MMVE software includes a mechanism that detects disconnections.

In the context of the continuous events approach, the *disconnection of a regular peer* can be handled as follows: After the MMVE software on a peer detects the disconnection from the network, the calculation of continuous events by the CEE is explicitly paused. In case the state of disconnection persists for only a short period of time, the calculation is resumed after the network connection is available again. After the calculation is resumed, for each executed continuous event the previously described mechanism for handling missed function calls that is part of the execution algorithm performs all missed function calls. In case the state of disconnection persists for a long period of time, the execution of continuous events by the CEE is stopped and the CEE terminates all executed continuous events. The CEM of the superpeer that manages the zone where the avatar of the disconnected regular peer

was located removes the peer id of the disconnected peer from all receiver lists. If the peer reconnects to the system after a long disconnection, a new AoI subscription is made by the peer according to the standard procedure of the target system. Based on this AoI subcription, the CEM is able to detect intersections between the AoI of the peer and the AoEs of existing continuous events and provides the reconnected peer with up-to-date copies of these continuous events for execution.

Analogical to peer crashes, the *disconnection of a superpeer* has implications for several parts of the P2P-MMVE system and, therefore, has to be handled by the overall system. A detailed elaboration of a concept and algorithms for handling the disconnection of a superpeer by the overall system is beyond the scope of this work. A concept based on a replication strategy like the concept that was outlined in the previous subsection is also a potential solution for handling disconnected superpeers. Assuming one or multiple copies of a superpeer are held by other regular peers with strong hardware capabilities, one of the copies is able to take over in case the original superpeer disconnects from the system. For continuous event support, the system has to make sure that the CEC of the new superpeer provides the CECs of all regular peers of the zone with its peer id in order to enable the sending of messages from the CECs of the regular peers to the CEC of the new superpeer via the DL.

4.3.4. Handling of Overloaded Peers

A P2P-MMVE system is run solely by user computers with very limited hardware and networking capabilities. The overloading of peers can be avoided to a certain degree by selecting peers with strong capabilities for system tasks, for example by using a superpeer model like the target system. Nevertheless, the calculation of continuous events for execution and management can potentially overload peers and, therefore, the case of overloaded peers has to be considered in the context of the continuous events approach. In the following, potential concepts for handling overloaded regular peers and superpeers are discussed.

The *overloading of a regular peer* can be handled by using a proxy concept. After the overloading of a regular peer by continuous event execution is detected, a proxy CEE is started on another peer with stronger hardware. This proxy CEE calculates the continuous events for the weak peer and sends single updates over the network to the overloaded peer. This can be realized in a straightforward way because artifical single events are calculated as part of of the continuous events approach. These

events can be calculated by the proxy CEE and then sent over the network to the overloaded peer. The drawback of a proxy concept is the creation of additional network overhead because the overloaded peer receives the state changes over the network instead of calculating them. In addition, the other peer can get overloaded by running the additional proxy CEE as well. This can be solved, for example, by the MMVE provider by adding own hardware to the peer-to-peer network. This hardware is then used to run the proxy CEEs.

Analogical to the discussions about superpeer crashes and disconnections, handling the *overloading of a superpeer* is strongly related to the superpeer concept of the target system and has to be handled by the overall P2P-MMVE system. If the superpeer of a zone gets overloaded, the previously described concept of replication can again be used. The overloaded superpeer is replaced by one of its copies with stronger capabilities. If none of the copies is strong enough to handle the load for a zone, a strategy that is often used by MMVEs with zones involves a dynamic retesselation and reassignment of superpeers to zones. The existing zone is split into several subzones and for each subzone a new superpeer is selected and assigned to the zone. This implies that the continuous events approach has to be extended by a mechanism that handles the split into zones for continuous event management. This can be handled as follows: At first, the managed continuous events are retrieved from the CEM of the original zone. After that, CEMs are started on the superpeers of the subzones. The retrieved continuous events then are assigned to the CEMs of the subzones based on intersections of their AoEs with the subzones. The CECs of the new superpeers of the subzones provide the CECs of all regular peers of their zones with their peer ids to enable the sending of messages from the CECs of the regular peers to the CECs of the superpeers via the DL. A strategy with a dynamic tesselation into subzones typically includes that subzones are merged into larger zones if the load of the superpeers of the subzones falls below a given threshold. This implies that the continuous events approach has to be extended with a mechanism that handles the merging of zones for continuous event management. This can be handled as follows: At first, all continuous events are retrieved from the CEMs of the subzones and merged into one set. Then the CEMs of the subzones are stopped, a CEM is started on the superpeer of the new zone and the set is handed over to the new CEM. The CEM starts managing the continuous events. The CEC of the superpeer of the new zone provides the CECs of all regular peers of the zone with its peer id in order to enable the sending of messages from the CECs of the regular peers to the CEC of the new superpeer via the DL.

5. Evaluation

This chapter presents an evaluation of the continuous events approach for a scalable propagation of continuous actions in P2P-MMVEs. In order to evaluate the continuous events approach, a software prototype was implemented and simulations of the resulting network information flow for the following types of continuous actions were performed: Movement of an object, movement of an object with a correlated spatial influence on the environment, movement of an object with multiple correlated spatial influences, and movement of object groups with multiple correlated spatial influences. In addition, reference values for an assessment of the results were created by running additional simulations for a P2P-MMVE with spatial publish subscribe but without continuous events and a client/server approach with basic dead reckoning.

This chapter is structured as follows: Section 5.1 discusses potential directions of an evaluation of the continuous events approach and gives reasons for the decision to focus on the network information flow. Section 5.2 describes the evaluation methodology. Sections 5.3 to 5.6 describe the performed simulations of the four varying types of continuous actions and the simulation results. Finally, Section 5.7 presents an assessment and discussion of the results.

5.1. Evaluation Focus

For an evaluation of the continuous events approach, two general directions come to mind. An evaluation of the networking aspects of the continuous events approach because the approach aims at a scalable propagation of continuous actions over the network in P2P-MMVEs. Or an evaluation of the calculation load that is put on the user hardware because the approach assumes that calculating the continuous actions on user computers is more beneficial than sending the information over the network. This implicitly includes the assumption that the calculation of continuous events is bearable by standard user computers.

In the past, studies showed that in highly interactive scenarios like online games the quality of user experience decreases for latencies higher than 100 ms [BCL+04]. Such scenarios typically have high demands on update propagation. Therefore, the given number of 100 ms represents a meaningful upper bound for P2P-MMVEs. Without the use of continuous events, a P2P-MMVE has to be able to propagate each of the single events representing a continuous action within 100 ms over the network to the target peer for a high quality of user experience. In a P2P-MMVE with continuous events, a peer has to be able to calculate each continuous event step that results in the creation of corresponding artificial single events at maximum within the same amount of time.

Preliminary experiments concerning the calculation load were performed by using code from the software prototype that is described later on. The experiments were run on an average office computer of the University of Mannheim (see Table B.1 in Appendix B for software and hardware specifications). The experiments were performed for all four types of continuous actions in combination with two alternatives of the continuous events approach with varying spatial modelings of AoEs. One of the continuous events approach alternatives used a minimum AoE and the other alternative a maximum AoE. Please see Subsection 5.2.3 for a description of the spatial modeling alternatives and Sections 5.3 to 5.6 for descriptions of the action types.

In the experiments, calculation steps were run for each type of continuous action and the number of steps that could be performed by the office computer within 100 ms was determined. In order to eliminate deviations caused by other processes on the office computer, the experiments each were repeated ten times and an average value was calculated.

The experiments showed that in 100 ms the office computer was able to calculate a huge number of operations for all four types of continuous actions and both spatial modeling alternatives. In the context of the continuous events approach alternative with a minimum spatial modeling, the office computer was able to calculate between 5723.6 and 6095.1 steps on average in 100 ms. In the context of the continuous events approach alternative with a maximum spatial modeling, the office computer was able to calculate between 5443.7 and 5948 steps on average in 100 ms. Please see Tables B.2 and B.3 in Appendix B for detailed results.

The results of the experiments indicate that processing power is not a limiting factor of the continuous events approach. The numbers of calculated steps that were

recorded in the experiments are high enough to assume that there are enough resources for calculating a very large number of continuous events. As a consequence of the experiments, it was decided that the focus of the evaluation is on the networking aspects of the continuous events approach.

5.2. Evaluation Methodology

In order to evaluate the potentially resulting network traffic of the continuous events approach when used for the propagation of varying types of continuous actions, the resulting information flow when using continuous events was simulated. The performed simulations aimed at getting results concerning the concept of continuous events and the continuous events approach in general. Therefore, the simulations focus on the basic continuous events approach without extensions as described in Section 4.2 and were performed for one zone using one CEM. This allows a precise evaluation of the general concept of continuous events and the potential of the continuous events approach without having to consider additional overhead for handling exceptional cases. In addition, because there is only one CEM in the system, the load for managing continuous events can be analyzed specifically based on this CEM.

5.2.1. Aspects and Metrics for Evaluation

The following aspects have to be evaluated for an assessment of the potential network traffic of a propagation of continuous actions with continuous events:

Network traffic reduction compared to basic P2P-MMVE system with spatial publish subscribe - The continuous events approach builds on a target P2P-MMVE system model that includes spatial publish subscribe as the main propagation mechanism. It aims at reducing the network traffic for propagation of continuous events in the context of this target system. Therefore it is important to evaluate the reduction in message number and payload by the continuous events approach in comparison to a basic P2P-MMVE with spatial publish subscribe propagation and no continuous events.

Network traffic reduction compared to MMVE system based on the client/server model with dead reckoning - State-of-the-art MMVEs typically use a client/server architecture and optimize the propagation of position changes of objects based on

dead reckoning. The implementation of a complete client/server-based MMVE for evaluation purposes is beyond the scope of this work. However, evaluations for a basic client/server approach with dead reckoning are performed in order to compare the continuous events approach with state-of-the-art systems. It is important to evaluate the performance of a peer-to-peer approach with spatial publish subscribe and continuous events in comparison to a basic client/server approach with dead reckoning concerning network messages and payload for propagation.

Influence of varying spatial modeling of the extent of continuous actions on the performance of the continuous events approach - As described earlier in this work, the continuous events approach enables a variable modeling of the spatial influence area of a continuous event for the initial propagation and the following management. In general, a minimum and a maximum modeling approach can be identified. Either the initial influence area of the continuous action is used (minimum approach), which theoretically should result in less messages for initial propagation but more overhead for management. Or the influence of all contained sub-actions of the continuous action is approximated by a large spatial area (maximum approach), which theoretically should result in more messages for the initial propagation but less messages for management. Intermediate types of modeling are also possible, but the minimum and maximum modeling approach mark the boundaries of modeling. Therefore, their influence on the performance is of most interest in the context of the evaluation of the continuous events aproach and these two alternatives are evaluated. It is important to evaluate the influence, potential advantages and disadvantages of a minimum and maximum modeling on the resulting network traffic in the context of the simulated types of continuous actions.

Comparison of average payload per message between dead reckoning and the continuous events approach - Although continuous events are able to carry more general informations and more complex actions than usual dead reckoning schemes, dead reckoning can be identified as the most direct related work to the continuous events approach. Therefore, it is interesting to examine the difference in size between the parameters that are needed for basic dead reckoning and the parameters for continuous events. Because of the more complex information that is contained in a continuous event the resulting payload should theoretically be larger in comparison to dead reckoning, which focuses on positions. The difference is evaluated in order to assess if the additional overhead is justified in consideration of the fact that a continuous event is able to carry much more complex continuous actions.

From the described aspects for evaluation, two metrics for assessing the results of the performed simulations can be derived: The potential *number of messages* that are sent and received for the propagation of continuous actions by the participating user computers. And the potential *payload* that is sent and received for the propagation of continuous actions.

5.2.2. Software Prototype

For the evaluation of the continuous events approach, a software prototype was implemented in Java Standard Edition 7. The prototype is able to simulate and record the potential network traffic of a P2P-MMVE with a spatial publish subscribe propagation and a client/server-based MMVE that uses areas of interest for interest management and propagation. In addition, it is able to simulate the management of continuous events over time and the resulting potential network traffic.

For the simulations, at first a set of movement data of user avatars and continuous actions was created for each type of continuous action. Then simulations for the varying systems were performed with the software prototype. After the simulation runs were performed, the recorded data about potential network traffic was retrieved and graphically prepared for presentation in this work.

For presentation purposes, a graphical display was added to the prototype. Figure 5.1 on page 110 presents a screenshot of the graphical user interface of the software prototype. The screenshot was taken during a simulation of continuous action type 2 with the CE Max AoE approach. More details about the specific action types and alternatives of the continuous events approach are given later on in this work. The interface displays the current simulation state including peer positions, AoIs and AoEs. By selecting one of the filtering options, the peer ids can be shown in addition to the positions and only AoEs of a certain peer can be shown.

The implemented *P2P-MMVE with spatial publish subscribe propagation* simulates a spatial publish subscribe functionality based on AoIs and AoEs as described in Section 2.3. The AoI subscriptions of the peers are updated over the course of a simulation run based on the stored movement data of user avatars. Continuous actions are propagated as corresponding single or continuous events to the AoEs. The simulator intersects the AoEs with the active AoI subscriptions. In case intersections are detected, an event is delivered to the corresponding peer, which means in the context of the simulation that the resulting network traffic and potential

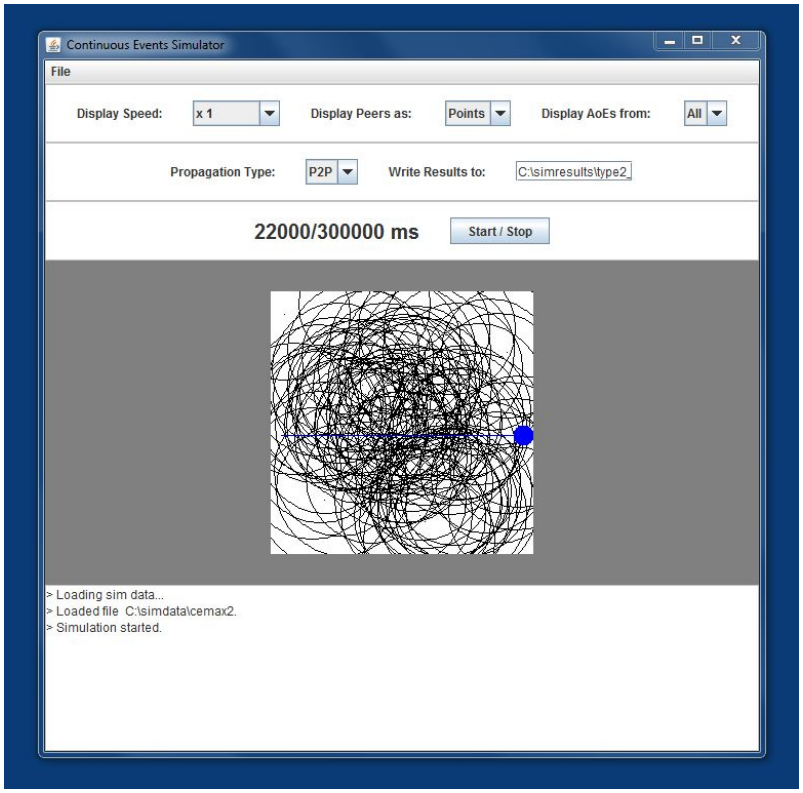


Figure 5.1.: Screenshot of the graphical user interface of the software prototype

payload of the parameters are recorded. For simulation of the continuous events approach, the software runs the CEM for the zone, simulates the management of existing continuous events and records the resulting network messages.

The implemented *client/server-based MMVE with dead reckoning* includes an interest management scheme based on the concept of area of interest (see Subsection 2.2.1). Each avatar is surrounded by an area of interest, which in the context of the simulations corresponds to the spatial AoI subscriptions. The areas of interest of avatars are updated based on the stored movement data over the course of a simulation run. In comparison to SPS, there are no complex spatial propagations. The state changes that result from continuous actions are propagated based on the position where they take place.

State-of-the-art client/server-based MMVEs use dead reckoning for the optimization of propagation of object movement (see Subsection 2.2.2). Therefore, the simulation of the client/server system also provides basic dead reckoning functionality. In case a continuous action results in an object that moves linearly towards a target location, dead reckoning is performed.

In contrast to the continuous events approach, which allows to aggregate movement and additional influences of a continuous action, in the client/server system the additional influence of moving objects has to be propagated by additional network messages. In client/server systems, state changes of several objects within an area are usually aggregated and sent as an aggregated message to clients. Therefore, in order to get more realistic results, the simulated client/server system includes a basic aggregation algorithm. In case an additional influence to the object movement takes place, a corresponding message is first sent to the server. The server determines all avatars or objects that are influenced and the state changes. After that, it determines the interest of clients in the state changes and provides all interested clients with an aggregated message.

For identification within the simulation, all simulated peers or client computers are assigned a *unique id*. This id has to be assigned dynamically by the system. In the Java prototype, such dynamically assigned system-wide unique ids were implemented based on the class `java.util.UUID` [Oraf]. A UUID consists of a 128-bit value. The basic concept of UUIDs is described by RFC 4122 [LMS]. The class `java.util.UUID` uses version 4 UUIDs, which are generated using a pseudo random number generator.

Continuous events were implemented as a Java class as follows: For identification within the system, a unique *continuous event id* is assigned dynamically to each continuous event. Continuous event ids were implemented based on `java.util.UUID`, analogical to peer or client ids. Timestamps, such as $t0$ or the points in time of the next and final step of a continuous event during execution and management, were implemented as numbers of type `java.lang.Long` [Orab]. Calculations with timestamps were implemented based on the class `java.util.Calendar` [Orad]. The interval between the steps δt and the number of overall intervals n were implemented as numbers of type `java.lang.Integer` [Oraa]. Because *function ids* are assigned before runtime, no dynamic creation of ids for functions is needed. Therefore, the prototype implements function ids as numbers of type `java.lang.Integer` and the ids are assigned as consecutive numbers. In Java, a number of type `java.lang.Integer` has a maximum positive value of 2.147.483.647. In order to get comparable results for the simulation of payload sizes (obviously the function with the id 1 creates less payload than a function with a higher number such as the maximum of 2.147.483.647), the simulation always records the worst case and uses the size of the number 2.147.483.647 when calculating the payload size. The parameters for the

aoe function f_{AoE} and effect function f_E were each implemented as arrays of type `java.lang.String` [Orac]. This allows to hand over parameters of varying types to the functions. In case the parameters include values of another data type than `java.lang.String`, implicit casts have to be implemented in the function code.

The *Continuous Event Controller*, *Continuous Event Executor* and *Continuous Event Manager* were implemented as seperate Java classes. All functionalities were implemented according to the given algorithms for basic continuous event support. The *receiver lists*, which are used by the Continuous Event Manager for storing the ids of all registered receiver peers for continuous events, were implemented based on the class `java.util.ArrayList`. All existing receiver lists of the Continuous Event Manager are stored for further use by the algorithms in a hashtable based on the class `java.util.Hashtable` [Orae] using the continuous event ids as keys.

The additional functionalities that are needed for continuous event execution and management were implemented as follows: The *Continuous Event Storage* was implemented as a hashtable based on the class `java.util.Hashtable` [Orae]. Continuous events can be stored and retrieved using their continuous event id as key. The *Timer Service* was realized as `ScheduledExecutorService` [Orag]. Because the distribution of function code for continuous events takes place before runtime, the information flow at runtime is not influenced. Therefore, the information flow created by the exchange of function code by the *Code Repository* is omitted and not simulated. An interface for functions was defined and functions for the action types were implemented as Java classes that implement the interface and added to the prototype.

5.2.3. Simulation Process

For the simulations, a data set containing the movement of avatars was created beforehand (see Subsection 5.2.5 for more details about the movement model). In addition, a data set of actions was created for each type of continuous action. To eliminate potential deviations because of movement patterns, all simulation runs used the same movement data in combination with the respective data set for the continuous action type. The continuous action types and characterization of the data sets are described in more detail in Sections 5.3 to 5.6.

For each combination of movement data and actions, simulations were performed with the software prototype for a peer-to-peer approach with spatial publish sub-

scribe propagation and no continuous events (P2P SPS), for a client/server approach with basic dead reckoning and areas of interest (CS DR), a peer-to-peer approach with spatial publish subscribe propagation and continuous events using a minimum spatial modeling of the influence (CE Min AoE), and a peer-to-peer approach with spatial publish subscribe propagation and continuous events using a maximum spatial modeling of the influence (CE Max AoE). The minimum and maximum spatial modelings that were used for the respective action types are explained in more detail later on in Sections 5.3 to 5.6.

The course of a simulation run by the software prototype can be described as follows: Over time, AoI subscriptions are updated based on the stored avatar movement data and the continuous actions from the action data set are propagated according to the given system approach. The information flow is only simulated and no data is sent over a real network interface. Instead, the potential network traffic is recorded. The number of messages that were sent and received over the course of the simulation run are recorded for each simulated user computer. In addition, the potential payload for each message is determined and recorded. In the context of the presented simulation, the potential payload corresponds to the size of all parameters that have to be carried by the message. The potential payload is determined by cummulating the size in bytes of all Java parameters that have to be sent in a certain message. This results in a large absolute payload size in comparison to existing real MMVE systems. However, because the software prototype was implemented in the same way for all simulated approaches, comparing the size in bytes between the approaches results in meaningful relative values.

5.2.4. Software and Hardware Specifications

All simulations were performed on a system with the following *software and hardware specifications*:

- Operating System: Windows Server 2007, 64 Bit
- Java Virtual Machine: Version 7, Update 7
- CPU: 2 x Intel Xeon QuadCore, 2.33GHz
- Memory: 6 GB

5.2.5. General Settings

Table 5.1 gives an overview of the general simulation parameters. Analogical to the evaluation of spatial publish subscribe in the system S-VON presented by Hu et al. [HWB+10], the settings emulate the conditions of Second Life. The *zone size* was set to 256 square meters, which corresponds to the size of a region in Second Life. A *user number* of 100 was used, which corresponds to the maximum number of users that can be handled by a Second Life region server. For interest management of the user avatars a circular *AoI* with a radius of 64 meters was used.

User movement was calculated based on the *Random Waypoint Model* (RWM) [JM96, BMJ+98]. User movement based on RWM can be described as follows: A user first selects a random target waypoint within the given zone. Then the speed for the movement towards that waypoint is selected from a uniformely distributed selection of speed values between a given minimum and maximum speed. After that, the user moves towards the waypoint with the determined speed. When the waypoint is reached, the user waits for a given pause time, which is determined from a uniformely selected distribution of values. When the pause time is over, a new waypoint and speed are randomly selected, the user moves towards the waypoint and so on.

For the presented simulations, the movement speed of user avatars was set to 5 meters per second, which roughly corresponds to the speed of running avatars in Second Life. The exact running speed in Second Life is defined as 5.13 meters per second [Linc]. When creating the movement data for the presented simulations, instead of defining a range from minimum to maximum speed for a random selection, minimum and maximum speed were set to the same value to get a constant speed for all users. In addition, the pause time was set to 0 seconds. This was done to eliminate potential influences of these factors on the simulation results.

Parameter	Value
Zone Size	256 x 256 meters
User Number	100
AoI Shape	Circle
AoI Radius	64 meters
Movement Model	Random Waypoint
Movement Speed Users	5 meters per second
Number of CEMs	1
Duration	5 minutes

Table 5.1.: Simulation parameters

As mentioned earlier, one CEM was used in order to allow a specific analysis of the potential network traffic for continuous event management. The performed simulation runs each had an overall duration of 5 minutes.

In the following sections, the simulated continuous action types are described and the results of the simulations are presented. Please note that the presented percentage numbers and numbers for payload size per message are rounded to two decimal places.

5.3. Action Type 1: Object Movement

5.3.1. Description

The first type of continuous actions that was simulated can be described as follows: As a result of a user action, an object in the virtual environment moves linearly from a given starting location A in the virtual environment towards a given target location B. The object changes its location for several times until the target location is reached. In the context of continuous action type 1, no further influences on other objects are correlated to the movement. Figure 5.2 illustrates continuous action type 1.

The action data for the simulation of continuous action type 1 was created based on the following parameters and assumptions: After every second, a user is selected randomly and a continuous action is triggered for this user. The moving object starts at one of 10 possible starting spots located at the western edge of the virtual environment. The starting spot is also selected randomly. The object moves an overall distance of 236 meters towards the eastern edge. It moves at a speed of 20 meters per second. The interval between each position change is 100 ms.

In a P2P-MMVE system with spatial publish subscribe and no continuous events

Figure 5.2.: Illustration of continuous action type 1

(P2P SPS), each position change triggers a new single event on the peer where the continuous action takes place. Each single event is then propagated to other peers based on the spatial influence. The AoE of a single event corresponds to a point located at the new position and a corresponding spatial publication is made. The publication is intersected with the available AoI subscriptions and the single event is provided to all peers with intersections.

In a client/server system with basic dead reckoning (CS DR), this type of continuous action results in the following information flow: The initial position of the object is sent in combination with the information needed to dead reckon the following position changes of the object (direction vector, speed). The informations are first sent from the client on which the continuous action took place to the server. The server then propagates the information to all clients whose AoI intersects with the initial position of the object. The target clients create object copies and, in the following, perform dead reckoning for the object movement. Over time, the server provides additional clients with an up-to-date position information, direction vector and speed of the object in case the object moves within their AoI or they move close enough to the object that their AoI intersects the current object position.

In a P2P-MMVE system with spatial publish subscribe and continuous events, each continuous action is represented by a continuous event. The continuous event is then propagated from the peer on which the continuous action took place to other peers by making one spatial publication. Over time, the CEM manages the continuous event, calculates the up-to-date AoE and provides additional peers with up-to-date copies in case their AoI intersects the current AoE of the continuous event. On the target peers, the continuous event is executed and moves the object via artificial single update events.

As described earlier, the following two spatial modeling alternatives were simulated: Propagation and management based on a minimum modeling of the influence (CE Min AoE) and a maximum modeling of the influence (CE Max AoE). Figure 5.3 on page 117 gives an overview of both alternatives for continuous action type 1. A minimum spatial modeling (left) uses an AoE that corresponds to the initial object position at location A. The continuous event is propagated initially to the point. For management, the CEM over time calculates AoEs corresponding to points that are located at the future object positions. A maximum spatial modeling (right) uses an AoE with the shape of a line that starts at location A, ends at location B and covers all future position changes. The continuous event is propagated initially to

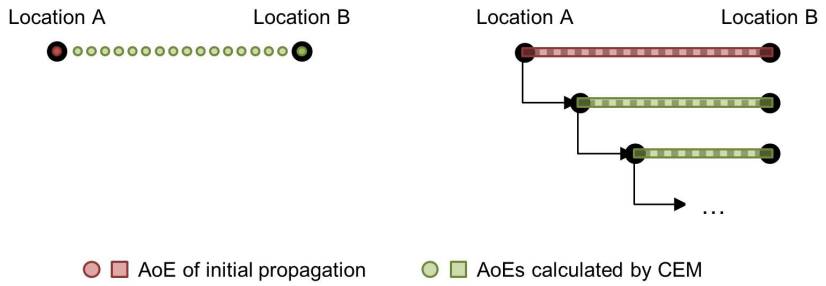


Figure 5.3.: Spatial modeling alternatives for continuous action type 1. The minimum alternative is shown on the left, the maximum alternative on the right.

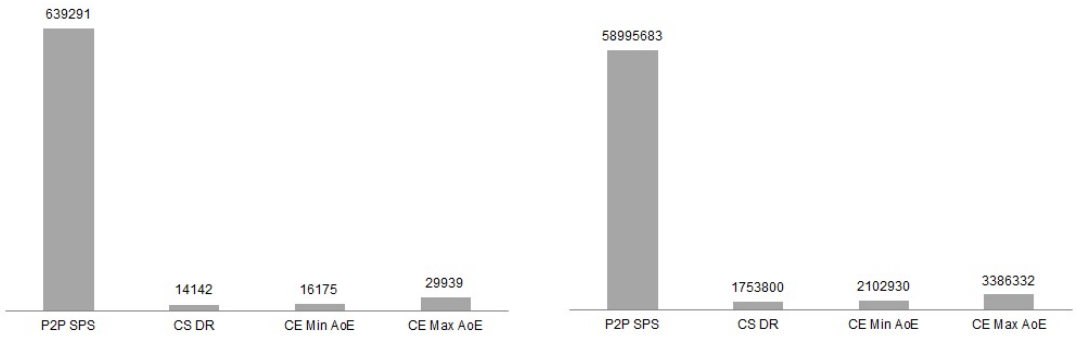
the maximum line. For further management, the CEM over time shrinks the line towards the target location B.

5.3.2. Simulation Results

Figure 5.4 on page 118 gives an overview of the overall results of the simulations of continuous action type 1. For the full results see Table A.1 in Appendix A.

Compared to P2P SPS without an optimized propagation of continuous actions, all simulated optimization approaches were able to reduce the overall number of messages for propagation as well as the payload significantly. CE Min AoE was able to reduce the messages by 97.47 percent and the payload by 96.44 percent. CE Max AoE reduced the messages by 95.32 percent and the payload by 94.26 percent. CS DR was able to propagate the continuous actions with 97.79 percent less messages and 97.03 percent less payload.

The average payload size per message for dead reckoning was 124.01 byte. The average payload size per continuous event message was slightly larger for both alternatives of the continuous events approach: 136.34 bytes for CE Min AoE and 136.58 bytes for CE Max AoE. The registration messages, which are used by the continuous events approach for the registrations of peers as receivers of continuous events, had a payload size of exactly 86 bytes per message for CE Min AoE as well as for CE Max AoE. It is plausible for all registration messages to have the same size because, no matter which alternative of the continuous events approach is simulated, all registration messages have to carry a peer id and a continuous event id. In the simulation prototype, both id types are implemented based on the same Java class `java.util.UUID` and the ids do not vary depending on the type of continuous



(a) Number of Messages Overall (b) Payload Overall (Byte)

Figure 5.4.: Overall simulation results for continuous action type 1

action or in case of an alternative spatial modeling. Therefore, the size has to be 86 bytes for all alternatives of the continuous events approach and all simulated continuous action types.

When comparing the overall results of the optimization approaches, CS DR shows the highest grade of reduction. A comparison of the spatial modeling alternatives indicates that CE Min AoE shows a higher reduction than CE Max AoE.

Figure 5.5 on page 119 shows a more detailed analysis of the information flow of the simulated optimization approaches. The figure backs up the observation that both simulated alternatives of the continuous events approach perform worse than the client/server approach with dead reckoning. CE Min AoE results in 14.38 percent and CE Max AoE in 111.70 percent more messages for propagation of continuous actions than CS DR. Concerning the payload, CE Min AoE resulted in 19.91 percent more payload and CE Max AoE in 93.09 percent more payload than CS DR.

Figure 5.5 further illustrates the varying characteristics of the information flow of the continuous events approach alternatives. CE Max AoE results in a higher absolute number of messages and payload size than CE Min AoE. The larger initial AoE results in more peers that are provided directly with continuous events instead of being provided with continuous event copies from the CEM. Therefore, more messages are sent directly between the peers when using the maximum spatial modeling. Because these peers have to register with the CEM as receivers, CE Max AoE results in more messages that are received by the managing peer that runs the CEM.

CE Min AoE needs less messages and payload than CE Max AoE. However, this comes at the cost of putting a higher burden on the managing peer. Less continuous events are propagated directly between regular peers because of the smaller initial

(a) Number of Messages Sent

(b) Number of Messages Received

(c) Payload Sent (Byte)

(d) Payload Received (Byte)

Figure 5.5.: Analysis of the information flow for continuous action type 1

AoE. These peers have to be provided with continuous event copies from the CEM over time. This leads to a more accurate provision with continuous events by the CEM based on an up-to-date calculated minimum AoE and saves messages. However, more continuous events are provided by the CEM. This leads to more network traffic on the sending side of the peer running the CEM.

5.4. Action Type 2: Object Movement with Single Additional Influence

5.4.1. Description

The second type of simulated continuous actions replicates the object movement pattern of the first action type, but adds an additional influence on surrounding objects to the end of the movement. Figure 5.6 on page 120 illustrates continuous

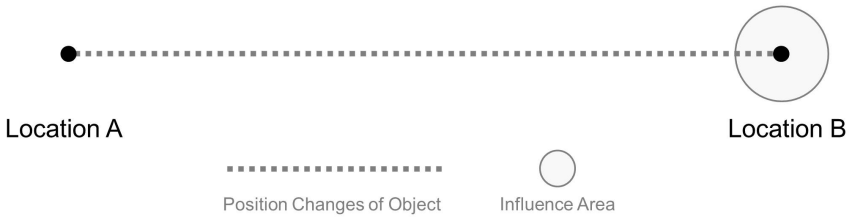


Figure 5.6.: Illustration of continuous action type 2

action type 2. As result of a user action, an object moves linearly from a starting position at location A towards a target location B. After arriving at location B, the object influences all user avatars whose position is located within a circular influence area surrounding location B.

The object movement data included in the action data for the simulation of continuous action type 2 was generated based on the same parameters and assumptions as the object movement data for continuous action type 1. A user is selected randomly every second and a continuous action is triggered for the selected user. A starting position is selected randomly from ten possible starting locations at the western edge of the virtual environment. Each object moves 236 meters towards the eastern edge at a speed of 20 meters per second. Position changes are propagated or calculated every 100 ms. In contrast to action type 1, an additional influence has to be added to each continuous action. After the object reached the target location, it influences all avatars within a circular area. The circle has a radius of 10 meters and is centered around the target location.

In a P2P-MMVE system with spatial publish subscribe and no continuous events (P2P SPS), all position changes of the object as well as the final influence at the target location have to be represented by single events. The position update events are propagated via spatial publications corresponding to points that are each located at the current position of the object. The final influence is propagated via a spatial publication to a circle that is centered around the target location and has a radius of 10 meters.

In a client/server system with basic dead reckoning (CS DR), the position changes of an object can be optimized based on dead reckoning, analogical to continuous action type 1. The impact of the final influence has to be propagated via additional messages. After the object reached the target location, the client that possesses the original object sends a message about the influence to the server. The server then

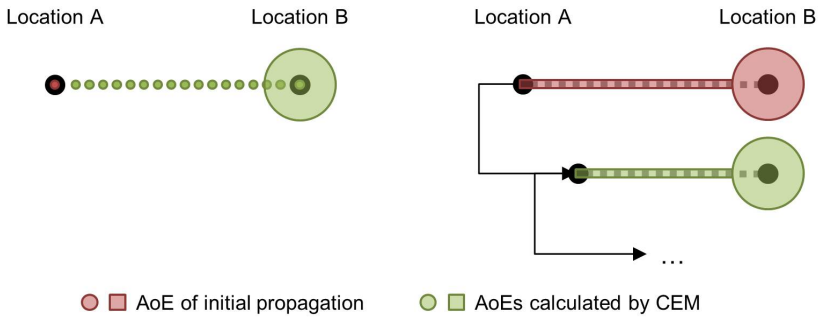


Figure 5.7.: Spatial modeling alternatives for continuous action type 2. The minimum alternative is shown on the left, the maximum alternative on the right.

determines all affected avatar objects, determines interest of clients and propagates the informations to the clients. As described earlier, the implementation of CS DR includes a basic message aggregation. The server provides all interested clients with a message that contains the effect and an aggregation of the ids of all affected avatar objects. The clients then update their copies of these avatars according to the aggregated message.

In a P2P-MMVE system with spatial publish subscribe and continuous events, the object movement and the final influence can be propagated by using one continuous event. No additional events are propagated for the final influences correlated to object movement. The future AoEs of the movement and the influence are calculated over time based on one common function. The effect of the movement and the influence are also calculated based on one common function.

Analogical to continuous action type 1, a minimum and a maximum spatial modeling alternative were simulated. Figure 5.7 gives an overview of the spatial modeling alternatives for continuous action type 2. The minimum alternative (CE Min AoE), which is displayed on the left, propagates the continuous events initially to a point at the starting location A. The CEM then over time calculates AoEs corresponding to points that are located at the respective object positions. In the end, a circular AoE corresponding to the final influence area is calculated. The maximum alternative (CE Max AoE), which is displayed on the right, propagates the continuous events initially to a maximum AoE that consists of a line for representing the influence of the object movement over time and a circle for representing the final influence. For further management, the CEM over time shrinks the maximum AoE towards the target location.

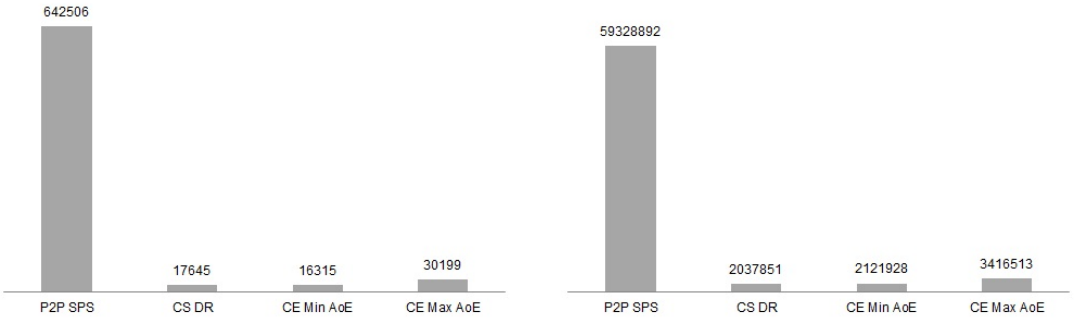
5.4.2. Simulation Results

Figure 5.8 presents the overall results of the simulations of continuous action type 2. For the full simulation results see Table A.2 in Appendix A.

Similar to the overall simulation results for continuous action type 1, all simulated optimization approaches were again able to reduce the overall number of messages as well as the payload significantly in comparison to P2P SPS. CE Min AoE reduced the messages by 97.46 percent and the payload by 96.42 percent. CE Max AoE lead to a reduction of messages by 95.30 percent and payload by 94.24 percent. CS DR was able to propagate the continuous actions of type 2 with 97.25 percent less messages and 96.57 percent less payload.

The reduction percentages of all approaches are similar to the percentages in the context of continuous action type 1. This is plausible because even there is an additional influence at the end of the object movement in the P2P SPS approach most events result from the position changes of the objects. The single events resulting from the final influence only make up a very small number of events. Therefore, the overall reduction percentages in the context of action types 1 and 2 are similar.

In the simulation of continuous action type 2, the average payload size of a message for dead reckoning was 124.01 bytes. A continuous event message in the simulation of CE Min AoE had an average payload size of 136.33 bytes. A continuous event message in the simulation of CE Max AoE had an average payload size of 136.58 bytes. For both alternatives of the continuous events approach, all messages for registering peers as receivers had a size of 86 bytes.



(a) Number of Messages Overall

(b) Payload Overall (Byte)

Figure 5.8.: Overall simulation results for continuous action type 2

(a) Number of Messages Sent

(b) Number of Messages Received

(c) Payload Sent (Byte)

(d) Payload Received (Byte)

Figure 5.9.: Analysis of the information flow for continuous action type 2

Figure 5.9 analyzes the information flow of the simulated optimization approaches in more detail. When confronted with continuous action type 2, CE Min AoE is able to beat CS DR concerning the overall number of messages: 7.54 percent less messages are needed compared to CS DR. However, CS DR performed slightly better than CE Min AoE concerning overall payload. Although CE Min AoE needed less messages, it resulted in 4.13 percent more payload compared to CS DR. A possible reason for this can be the slightly larger payload of continuous event messages in comparison to dead reckoning messages. Concerning the metrics overall number of messages and payload, CE Max AoE showed the worst performance of the three optimization approaches.

When comparing the information flow of CE Min AoE and CE Max AoE, the results show similar characteristics as the results for action type 1. The large AoE of CE Max AoE results in more messages and payload that are sent by the regular peers. Because of the large initial AoE, more peers are provided directly with continuous

events after their creation. Less continuous events have to be provided by the CEM afterwards. CE Min AoE propagated the continuous actions with less overall messages and payload than CE Max AoE, but results in more messages and payload sent by the peer that runs the CEM. The small initial AoE of CE Min AoE provides less peers directly with continuous events after their creation. More continuous events have to be provided by the CEM. The better general performance of the minimum alternative hints at a higher grade of accuracy of this alternative. Because of the travel time of the object, the AoI subscriptions of peers initially intersect parts of the maximum AoE, but have moved away from the continuous action until the object arrives. The minimum alternative does not provide these peers with the continuous event.

5.5. Action Type 3: Object Movement with Multiple Additional Influences

5.5.1. Description

Continuous action type 3 also builds on the object movement pattern of the previous action types. Figure 5.10 on page 125 illustrates the characteristics of continuous action type 3. A continuous action of type 3 results in an object that moves linearly from a starting position at location A towards a target location B. In contrast to the other types, the object influences avatars in its surroundings for multiple times. Whenever the object reaches a new position, it influences avatars within an influence area surrounding its current position. Similar to the single final influence of continuous action type 2, the influences of continuous action type 3 over time are modeled by circles. With each move, an object influences all avatars within a given circular area surrounding the respective position of the object at that time. Please note that for reasons of simplification, the figure shows more dots indicating position changes than influence areas. However, in the simulation the influence area moves together with the object. Therefore, with every position change there is also a new influence area.

The object movement data for continuous action type 3 again was created based on the same parameters and assumptions as the data that was created for the previous action types. A continuous action is triggered every second for a randomly selected user. The starting position of the object is chosen randomly from ten locations at the

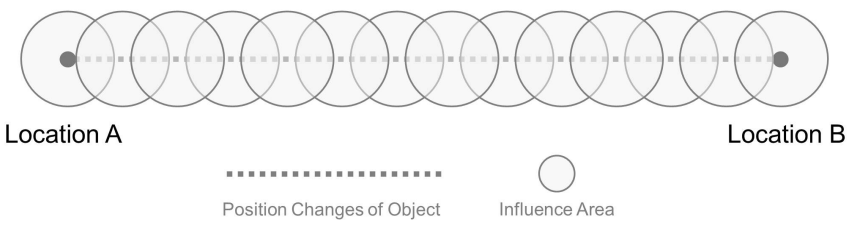


Figure 5.10.: Illustration of continuous action type 3

western edge of the virtual environment. The object moves 236 meters towards the eastern edge at a speed of 20 meters per second. The interval between propagation or calculation of position changes is 100 ms. In contrast to the action data of the previous types of continuous actions, a spatial influence was added after each position change of an object. The object influences all avatars within a circle that is centered at the new position and has a radius of 10 meters.

In a P2P-MMVE with spatial publish subscribe and no continuous events (P2P SPS), all position changes of objects and all additional influences have to be represented by single events. The position update events are propagated via spatial publications of the shape of points that are located at the respective position of the object. The additional influences are propagated via spatial publications with a circular shape centered around the position and a radius of 10 meters.

In a client/server system with basic dead reckoning (CS DR), the position changes of the objects of continuous action type 3 can also be optimized by dead reckoning, analogical to the object movement of the previous action types. The additional influences that are triggered by the object movement over time all have to be propagated via additional messages. Analogical to continuous action type 2, the client that possesses the original object first sends a message describing the influence to the server. The server then determines the affected avatar objects, determines interest and propagates an aggregated message to all interested clients.

A P2P-MMVE system with spatial publish subscribe and continuous events is able to represent all influences, all position changes and all influences over time of a moving object by one continuous event. No additional events are needed for the influences. All future AoEs of the position changes and the influences are calculated based on one common function. In addition, the effect of the position changes and the influences are also calculated based on one common function.

Figure 5.11 on page 126 shows the minimum and maximum spatial modeling al-

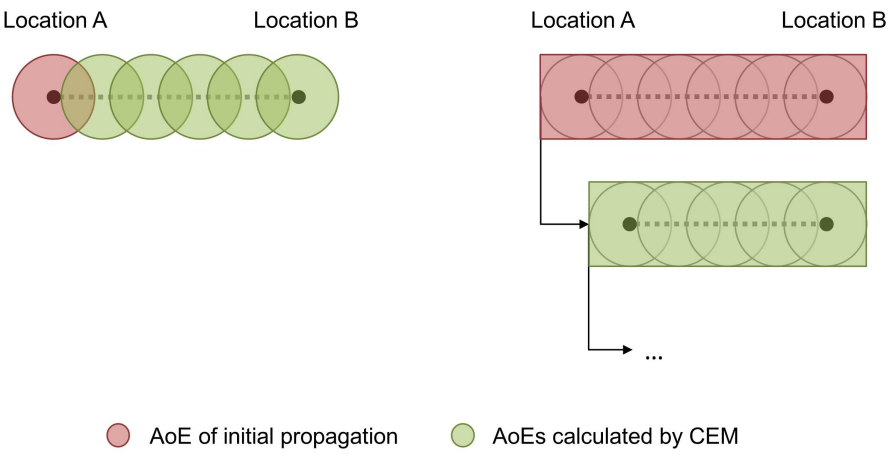


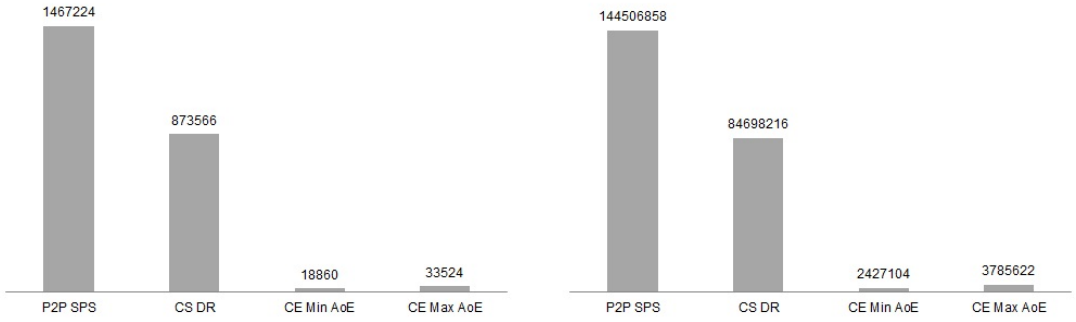
Figure 5.11.: Spatial modeling alternatives for continuous action type 3. The minimum alternative is shown on the left, the maximum alternative on the right.

ternatives for continuous action type 3. The minimum alternative (CE Min AoE), which is displayed on the left, initially propagates each continuous event to a circular AoE with a radius of 10 meters whose center point is located at the starting location of object movement. The CEM then over time recalculates the location of the circular AoE according to object movement. Please note that the position of the object is always covered by the circle. Therefore, for the initial propagation and the following management no additional AoE with the shape of a point has to be calculated. The maximum alternative (CE Max AoE), which is displayed on the right, uses a rectangle to approximate all circular influences and all positions over time. Each continuous event is initially propagated to the maximum rectangle. The CEM then over time shrinks the size of the rectangle towards the target location.

5.5.2. Simulation Results

Figure 5.12 on page 127 presents the overall results of the simulations of continuous action type 3. For the full simulation results see Table A.3 in Appendix A.

Compared to P2P SPS, again all simulated optimization approaches were able to reduce the overall message number and payload. However, for continuous action type 3 both alternatives of the continuous events approach showed a significantly better performance than CS DR. CE Min AoE reduced the messages by 98.71 percent and the payload by 98.32 percent. CE Max AoE reduced the messages by 97.72 percent and the payload by 97.38 percent. CS DR needed 40.46 percent less messages and 41.39 percent less payload than P2P SPS.



(a) Number of Messages Overall

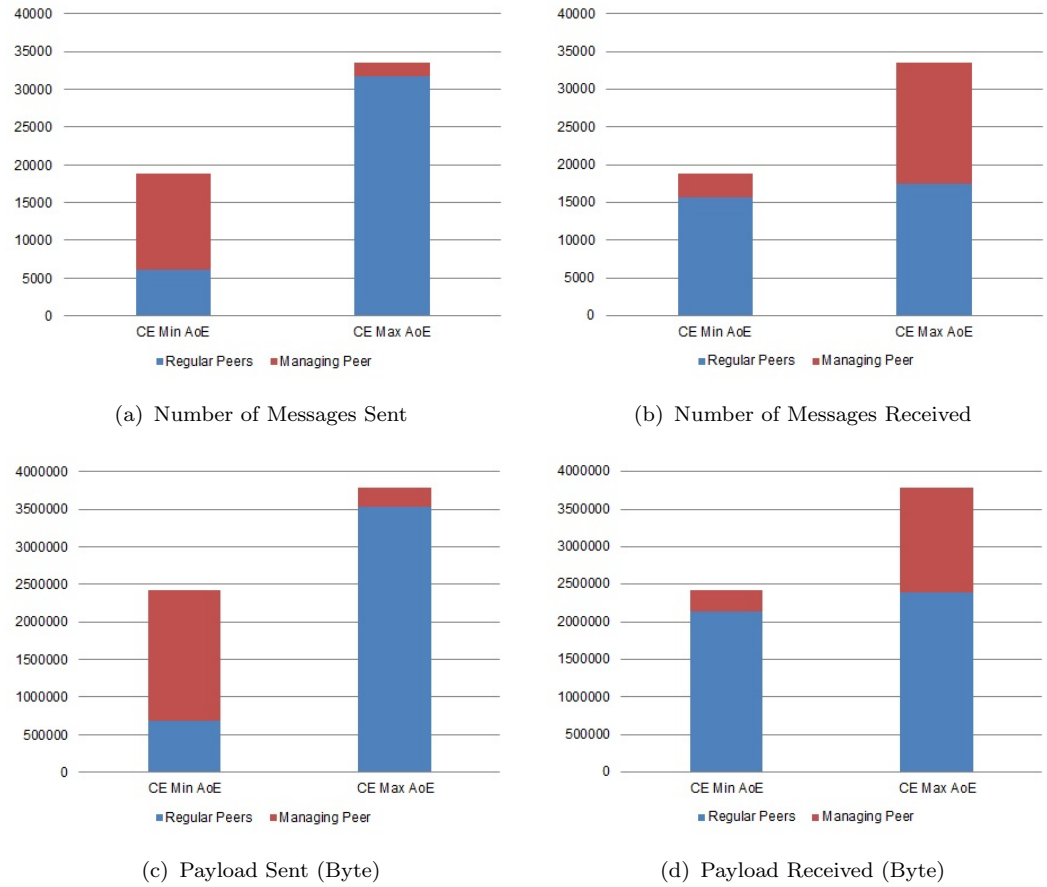
(b) Payload Overall (Byte)

Figure 5.12.: Overall simulation results for continuous action type 3

The overall results presented in Figure 5.12 indicate that for action type 3 the continuous events approach has a clear advantage over the simulated client/server approach. Although the client/server approach optimizes the propagation of the spatial influences by using the previously described aggregation mechanism, it was not able to beat or at least produce similar results to the results of the simulated alternatives of the continuous events approach.

In the simulations of continuous action type 3, dead reckoning messages had an average payload of 124.01 bytes. Although the additional informations about multiple influences were propagated and applied in addition to the object movement by the continuous events, the size of the average payload per continuous event was almost identical to the continuous events that were used for the previous action types. Continuous event messages for CE Min AoE had an average payload of 136.34 bytes and continuous event messages for CE Max AoE had an average payload size of 136.58 bytes. The payload size per registration message by the peers was again 86 bytes for both alternatives of the continuous events approach.

The reason for the nearly identical payload of continuous events compared to the previous action types can be identified in the design of the continuous events approach. Although the additional spatial shapes and their effect have to be calculated, only function ids have to be sent. Function ids had a constant size in the performed simulations under the assumption of the worst case of the highest Integer value. In addition, for both modeling alternatives only the parameter values describing location A and B have to be included in the continuous event. The initial circle and the following circle of the minimum alternative as well as the rectangles of the maximum alternative can be calculated based on this information. In comparison, for continuous action types 1 and 2, the functions also needed the parameter values



(a) Number of Messages Sent

(b) Number of Messages Received

(c) Payload Sent (Byte)

(d) Payload Received (Byte)

Figure 5.13.: Analysis of the information flow for continuous action type 3

describing location A and B. Therefore, all additional influences can be calculated without creating a significant payload overhead.

In summary, the overall results hint at a strong advantage of the continuous events approach for the propagation of complex continuous actions like action type 3 over the simulated state-of-the-art approach based on the client/server model with dead reckoning and message aggregation by the server.

Figure 5.13 presents a more detailed analysis of the information flow of the simulated alternatives of the continuous events approach. The information flow of the continuous events approach alternatives again shows similar characteristics than the information flow that was observed in the previous simulations. CE Min AoE was able to reduce the overall message number and payload stronger than CE Max AoE. However, this came at the cost of a higher network load on the sending side of the peer that runs the CEM. On the other hand, the large rectangular AoEs of CE Max AoE led to more continuous events that were propagated directly between regular

peers. The resulting information flow was more distributed over the sending sides of the network connections of the regular peers and less network load was on the managing peer.

5.6. Action Type 4: Object Group Movement with Additional Influences

5.6.1. Description

Continuous action type 4 was designed to simulate the performances of the varying optimization approaches when confronted with very complex continuous actions that include the movement of multiple objects and a high number of additional correlated influences on objects in the virtual environment. A continuous action of type 4 includes ten objects that move through the virtual environment. Each of these objects results in multiple additional influence areas. An example for such a type of action is the cloud example given in Section 3.3. A peer is assigned the task of controlling the display and of applying the effect of multiple cloud objects on other peers. All simultaneously created clouds can be represented and propagated by using one continuous event.

Figure 5.14 on page 130 illustrates the characteristics of continuous action type 4. A continuous action results in multiple objects that move linearly from their starting positions (locations A, C, E, G) towards given target locations (locations B, D, F, H). On their way through the virtual environment, after each position change the objects influence avatars within a circular influence area centered at their current positions. Please note that similar to figure 5.10 for reasons of simplification the figure shows more dots indicating position changes than influence areas. Nevertheless, in the created action data each position change results in a new influence area. In addition, for reasons of simplification the figure also shows only four objects while the continuous actions included in the action data include ten objects.

The objects included in continuous action type 4 all move according to the movement pattern from the previous action types. Each object starts at the western edge of the virtual environment and then moves linearly 236 meters towards the eastern edge at a speed of 20 meters per second. The ten starting spots that were used for random selection by the previous action types depict the starting spots of the ten moving objects included in action type 4. The interval between propagation or calculation

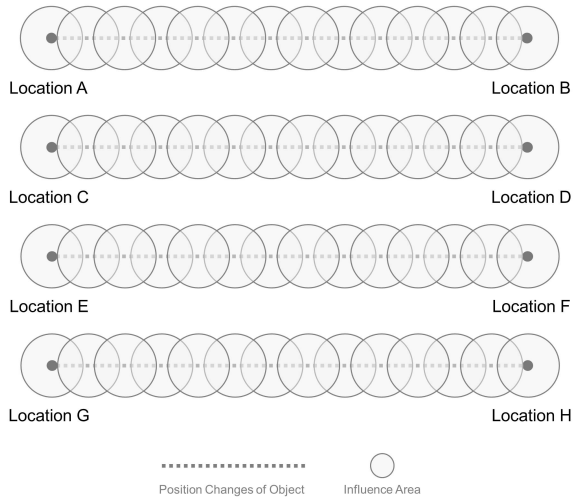


Figure 5.14.: Illustration of continuous action type 4

of position changes is 100 ms. The influence areas that are correlated to the position changes of objects are circular and have a radius of 10 meters. The creation of the action data set for continuous action type 4 was performed analogical to the data sets for the previous action types. After every second, a peer was selected randomly and a continuous action of type 4 was triggered for this peer.

In a P2P-MMVE with spatial publish subscribe and no continuous events (P2P SPS), all position changes of the objects and all additional influences result in single events that are propagated via spatial publications between the peers. The position updates are propagated to spatial publications corresponding to points. The events for the additional influences are propagated to spatial publications corresponding to circles with a radius of 10 meters and centered at the object position.

In a client/server system with basic dead reckoning (CS DR), the movement of the objects can be optimized by dead reckoning, similar to the previous action types. However, dead reckoning is usually performed in dependency of an object. Therefore, in the context of continuous action type 4 the informations for dead reckoning have to be sent individually for each of the ten objects that are contained in a continuous action. In addition, for each moving object all influences of the objects over time have to be propagated by additional messages. The propagation of influences works analogical to the propagation that was described earlier for continuous action types 2 and 3. The client that possesses the original object first sends a message about the influence to the server. The server then determines the influenced avatars,

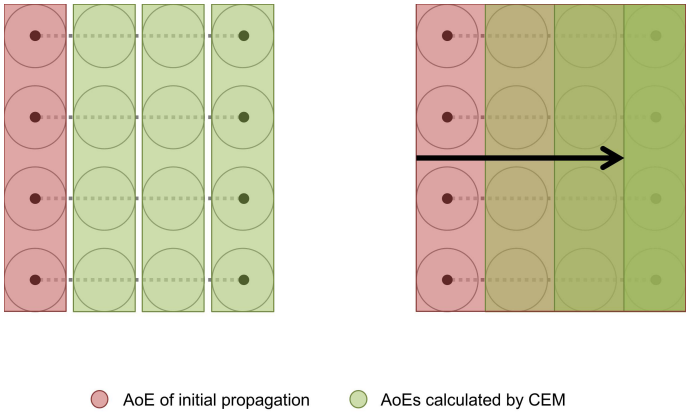


Figure 5.15.: Spatial modeling alternatives for continuous action type 4. The minimum alternative is shown on the left, the maximum alternative on the right.

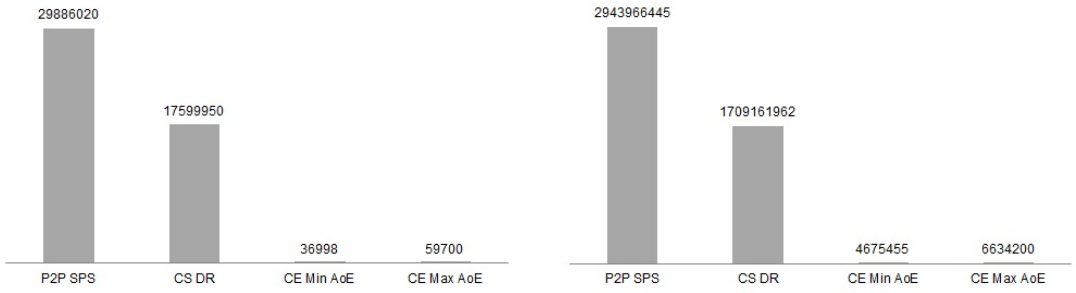
determines interest and provides all interested clients with an aggregated message.

In a P2P-MMVE with spatial publish subscribe and continuous events, the movement of all ten objects and all influences of these objects over time can be represented by one continuous event. Figure 5.15 gives an overview of the spatial modeling alternatives for continuous action type 4. The minimum alternative (CE Min AoE), which is displayed on the left, initially propagates the continuous events to a rectangle that approximates all initial circular influences of the objects. The object positions are covered by the rectangle as well. Therefore, no additional AoEs have to be calculated for the position changes. For further management of the continuous event by the CEM, the corner points of the rectangle are then moved over time according to the movement of the objects. The maximum alternative (CE Max AoE), which is displayed on the right, initially propagates the continuous events to a very large rectangle that approximates all circular influences of all objects over time. Again, no additional AoEs are needed for the position changes because the rectangle always covers all object positions. For further management, the maximum rectangular AoE is then shrinked over time according to the movement of the objects.

5.6.2. Simulation Results

Figure 5.16 on page 132 presents the overall results of the simulations of continuous action type 4. For the full simulation results see Table A.4 in Appendix A.

For continuous action type 4, all optimization approaches again were able to reduce the overall number of network messages and payload for propagation of continuous



(a) Number of Messages Overall

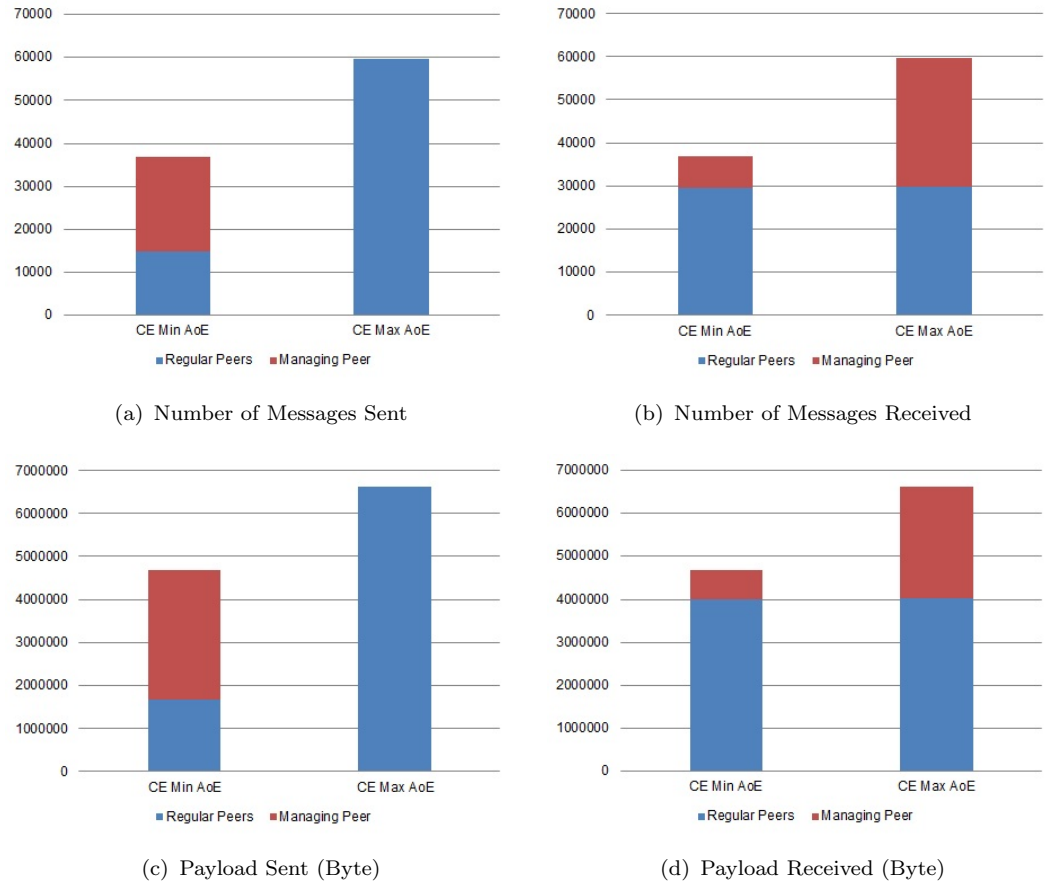
(b) Payload Overall (Byte)

Figure 5.16.: Overall simulation results for continuous action type 4

actions in comparison to the basic peer-to-peer approach P2P SPS. When confronted with very complex continuous actions like action type 4 both continuous events approach alternatives clearly have an advantage compared to CS DR. CE Min AoE and CE Max AoE resulted in a significantly higher message and payload reduction than CS DR. Compared to P2P SPS, CE Min AoE reduced the messages by 99.88 percent and the payload by 99.84 percent. CE Max AoE resulted in a message reduction of 99.80 percent and a payload reduction of 99.84 percent. Although CS DR was performing worse than CE Min AoE and CE Max AoE, the implemented basic dead reckoning algorithm and aggregation technique of CS DR were at least able to reduce the messages by 41.11 percent and the payload by 41.94 percent in comparison to P2P SPS.

Although the movement and influences of ten moving objects were propagated per continuous event, the average size of the payload per continuous event message was not significantly larger than the average size of the payload per message for the previously simulated action types. For CE Min AoE, a continuous event message had an average payload size of 136.27 byte. For CE Max AoE, a continuous event message had an average payload size of 136 byte. Registration messages again had an average payload of 86 bytes for both continuous events approach alternatives. In the simulation of CS DR, a dead reckoning message had an average payload size of 123.00 bytes.

Figure 5.17 on page 133 analyzes the information flow of the simulated alternatives of the continuous events approach. Analogical to the simulations for the previous three continuous action types, CE Min AoE performed better concerning the absolute number of messages and payload but puts a higher burden on the peer that runs the CEM. In comparison to CE Max AoE, more continuous events had to be provided



(a) Number of Messages Sent

(b) Number of Messages Received

(c) Payload Sent (Byte)

(d) Payload Received (Byte)

Figure 5.17.: Analysis of the information flow for continuous action type 4

by continuous event management after the initial propagation. On the other hand, CE Max AoE results in more messages and payload that are sent by regular peers because more continuous events are propagated directly between the regular peers via the initial propagation of continuous events. The information flow of CE Max AoE hints at a network load that is more distributed over the peers in comparison to the information flow of CE Min AoE.

The presented information flow of CE Max AoE shows that CE Max AoE not only resulted in less messages and payload for provision of existing continuous events to peers by continuous event management, but rather in no messages at all for provision of existing continuous events. A potential reason is the very large size of the rectangle that is used for initial propagation by CE Max AoE. This rectangle is nearly as large as the given size of the virtual environment. Therefore, it seems that when using the maximum spatial modeling for simulation all peers were provided initially with the continuous event from the peer where the continous action took

place. As a result, in the simulation of continuous action type 4 with CE Max AoE no more continuous event copies were sent from the managing peer with the CEM to regular peers. The managing peer only received registration messages by the peers that were initially provided with the continuous events. In combination with the observed high overall reduction rate of CE Max AoE compared to a basic P2P SPS system, this observation indicates that for continuous action type 4 the continuous events approach is able to reduce the network traffic significantly, even the continuous events are propagated simply to the whole zone instead of using a sophisticated approximation of the influence.

5.7. Assessment and Discussion of Results

The previous sections described the performed simulations and presented their results. This section gives an assessment and discussion of the results in relation to the aspects for evaluation that were identified in Subsection 5.2.1.

Network Traffic Reduction Compared to Basic P2P-MMVE System with Spatial Publish Subscribe

Table 5.2 on page 135 presents a concluding overview of the reduction percentages concerning overall message number and payload that were calculated based on the recorded simulation results.

The results presented by Table 5.2 show that the continuous events approach is able to reduce the message number and payload for the propagation of continuous actions significantly in comparison to a basic peer-to-peer approach with spatial publish subscribe and no optimization by continuous events. For all simulated types of continuous actions, both simulated spatial modeling alternatives were able to reduce the messages and payload for propagation of continuous actions significantly by at least 94.24 percent. When comparing the percentage numbers for message number and payload, it can be observed that the continuous events approach led to a slightly higher reduction rate concerning the message number than the payload.

The results further show that the reduction rate increases for complex continuous actions that involve multiple objects and influences per continuous action. The reduction rates for continuous action types 3 and 4 are clearly higher than the rates

	Action Type 1	Action Type 2	Action Type 3	Action Type 4
-CS DR-				
Message Reduction	97.79	97.25	40.46	41.11
Payload Reduction	97.03	96.57	41.39	41.94
-CE Min AoE-				
Message Reduction	97.47	97.46	98.71	99.88
Payload Reduction	96.44	96.42	98.32	99.84
-CE Max AoE-				
Message Reduction	95.32	95.30	97.72	99.80
Payload Reduction	94.26	94.24	97.38	99.84

Table 5.2.: Reduction percentages of the simulated optimization approaches

for continuous action types 1 and 2. When confronted with type 4, the most complex type of continuous action, both spatial modeling alternatives of the continuous events approach resulted in the highest reduction percentage. They reduced the messages by more than 99.80 percent and the payload by 99.84 percent.

This general trend of an increasing reduction rate by the continuous events approach with an increasing complexity of continuous actions is only broken by a comparison of the reduction percentages for continuous action types 1 and 2. Continuous action type 2 is more complex than continuous action type 1 because it includes an additional influence. However, the reduction percentages for continuous action type 2 are slightly lower than for continuous action type 1. A potential explanation for this observation is the circular influence area at the end of the object movement in continuous action type 2. In comparison to continuous action type 1, this results in an overall larger AoE of the corresponding continuous event. The larger AoE for initial propagation has potentially more intersections with AoIs and, therefore, eventually results in more initial propagations of continuous events.

In summary, the simulation results show that for the simulated types of continuous actions the continuous events approach was able to reduce the overall message number and payload for propagation of continuous actions strongly in comparison to a basic P2P-MMVE system with spatial publish subscribe.

The observed reduction percentages of more than 90 percent are high enough to justify the conclusion that the continuous events approach has the general potential to enhance the overall scalability of a P2P-MMVE system and to propagate continuous actions in a scalable way.

Network Traffic Reduction Compared to MMVE System based on the Client/Server Model with Dead Reckoning

The results presented in Table 5.2 further allow to compare the performance of the continuous events approach to CS DR, the simulated version of a MMVE system based on the client/server model with basic dead reckoning.

For continuous action types that involve only object movement, for example continuous action type 1, the implemented version of dead reckoning resulted in a higher reduction rate in percent concerning the number of messages and payload than both alternatives of the continuous events approach. The message number was reduced by 97.79 percent compared to 97.47 percent (CE Min AoE) and 95.32 (CE Max AoE). The payload was reduced by 97.03 percent compared to 96.44 (CE Min AoE) and 94.26 percent (CE Max AoE). These percentage numbers imply that the concept of dead reckoning can not be beaten by the continuous events approach for propagating continuous actions that involve only movement.

However, the basic concept of dead reckoning is tailored specifically to optimize the propagation of movement. Considering the fact that the continuous events approach aims at supporting the propagation of generic types of continuous actions, the recorded results indicate a good potential of the continuous events approach because the continuous events approach alternative CE Min AoE was at least able to come close to CS DR.

For continuous action types that predominantly involve object movement, for example continuous action type 2, CS DR resulted also in a higher reduction rate in percent than both alternatives of the continuous events approach concerning overall payload. The payload was reduced by 96.57 percent compared to 96.42 percent (CE Min AoE) and 94.24 percent (CE Max AoE). However, CS DR did not result in a generally higher reduction rate concerning the overall number of messages. One of the continuous events approach alternatives, CE Min AoE, was able to beat CS DR with a reduction rate of 97.46 percent compared to a reduction rate of 97.25 percent by CS DR.

Continuous action types 3 and 4 are more complex than continuous action types 1 and 2 because they involve multiple additional influences that are correlated to object movement. For these types of continuous actions, CS DR lead to a significantly lower reduction rate in percent than the continuous events approach. The reduction rate of both continuous events approach alternatives remained significantly above

90 percent concerning overall message number and payload. The overall number of messages and payload were reduced by at least 97.38 percent. In contrast, the reduction rate of CS DR dropped to below 50 percent. The overall message number and payload were reduced by at least 40.46 percent and at most 41.94 percent. Basic dead reckoning concepts can not be used besides object movement. Therefore, in CS DR additional messages have to be sent to propagate the resulting state changes of the additional influences. Although the simulated version of CS DR included an aggregation mechanism for optimizing the propagation of these state changes, the additional messages resulted in a decreasing reduction rate in comparison to the continuous events approach.

In summary, a comparison of the reduction rates in percent between the continuous events approach and the simulated version of a client/server system with basic dead reckoning shows that with increasing complexity of the continuous action the advantage of the continuous events approach increases. Dead reckoning aims at optimizing the propagation of movement while the continuous events approach aims at optimizing the propagation of continuous actions in general. Therefore, CS DR has an advantage for continuous actions that only or predominantly involve object movement. The continuous events approach has an advantage for complex continuous actions with additional interactions and influences on other objects in the virtual environment.

Influence of Varying Spatial Modeling of the Extent of Continuous Actions on the Performance of the Continuous Events Approach

The presented results indicate that there are correlations between varying spatial modeling alternatives for the extent of continuous events and the resulting network traffic reduction as well as the characteristics of the resulting information flow.

In all performed simulations, the use of a minimum spatial modeling by CE Min AoE resulted in a higher message and payload reduction than the use of a maximum spatial modeling by CE Max AoE. However, the difference between the reduction percentage of CE Min AoE and CE Max AoE gets smaller with increasingly complex continuous events. For continuous action type 1, CE Min AoE reduced the overall message number by 2.15 percent more and the overall payload by 2.18 percent more than CE Max AoE. For the most complex type of continuous action, type 4, CE Min AoE reduced the overall message number only by 0.08 percent more than CE Max AoE. The reduction rates concerning the overall payload were even identical.

Concerning the characteristics of the information flow, in all performed simulations the minimum spatial modeling resulted in less overall messages and payload but at the same time led to a higher network load for the peer that runs the CEM. Because of the smaller initial AoE of the minimum modeling, less continuous events were propagated directly between peers and more peers had to be provided with existing continuous events by the CEM. On the other hand, the maximum spatial modeling resulted in more overall messages and payload than the minimum modeling, but the network load is more distributed over the peers. Because of the large initial AoE, more continuous events were propagated initially between the peers and, as a result, the managing peer had to carry less network load.

This observation backs up the decision by the continuous events approach to support several spatial modelings and shapes instead of deciding about a certain spatial modeling for the extent of continuous actions. Depending on the type of continuous action, the use case and target system, the spatial modeling can be chosen accordingly. By selecting a certain spatial modeling, the characteristics of the information flow can be directly influenced. In case there is a superpeer with strong networking capabilities or the MMVE consists of small zones, a minimum modeling seems to be the best choice because of the highest reduction rate. In case the superpeer has to carry less networking load or the MMVE consists of large zones, a maximum modeling seems to be a feasible choice as well because it leads to more directly propagated continuous events and the reduction rate is also high.

Comparison of Average Payload per Message between Dead Reckoning and the Continuous Events Approach

Table 5.3 on page 139 presents a comparison of the average payload sizes in bytes per message that were calculated based on the recorded results of the simulations.

The comparison shows that independently from the simulated type of continuous action the payload size of continuous events remained similar. Even for continuous actions with increasing complexity, the average payload size per continuous event did not increase. This backs up the decision to include function ids instead of code into continuous events. Because the code for the complex manipulation of multiple object states is not contained in the continuous events that are sent at runtime, it is possible to keep the average payload size relatively constant, even for very complex continuous actions.

	Action Type 1	Action Type 2	Action Type 3	Action Type 4
Dead Reckoning	124.01	124.01	124.01	123.00
CE Min AoE	136.34	136.33	136.34	136.27
CE Max AoE	136.58	136.58	136.58	136.00

Table 5.3.: Average payload size in bytes per messsage of the simulated optimization approaches

In all simulations, continuous events had a larger average payload than messages that included the parameters for dead reckoning. When using CE Min AoE, the average payload of continous events was 12.33 bytes larger for continuous action types 1 and 3, 12.32 bytes larger for continuous action type 2, and 13.27 bytes larger for continuous action type 4. When using CE Max AoE, the average payload of continuous events was 12.57 bytes larger for continuous action types 1, 2 and 3, and 13 bytes larger for continuous action type 4.

This shows that continuous events are able to support a scalable propagation of generic, potentially very complex continuous actions with only a slightly larger payload than dead reckoning that only supports movement. In contrast to dead reckoning, the continuous events approach needs additional registration messages. However, as shown in the previous discussion about the general performance of the approaches, the continuous events approach is able to propagate continuous actions in a scalable way even including the registration messages.

When comparing the average payload sizes that are presented in Table 5.3 it can be observed that the average payload size per dead reckoning message is identical for continuous action types 1 to 3. The average payload size for continuous action type 4 deviates and is smaller. A similar pattern can be observed for the average payload sizes per continuous event. A potential reason for this is the reuse of the object movement patterns for the first three types of continuous actions. For continuous action type 4, only the parameter values for movement were reused because in contrast to the other three action types ten objects move at a time instead of one randomly selected object. However, because this observation can be made for all simulated optimization approaches and all approaches are affected in a similar way, the results are nevertheless meaningful.

6. Conclusion and Future Work

This final chapter gives a summary of this work in Section 6.1. Then Section 6.2 classifies this work in the context of the presented related work. Section 6.3 assesses the fulfilment of requirements by the presented approach. Finally, Section 6.4 gives an outlook on potential directions of future research.

6.1. Summary

Chapter 1 first gave an overview of the research objectives of this work. This work aims at finding an approach to send the outcome of continuous actions in a scalable way over the network of a P2P-MMVE. It puts an emphasis on designing a system architecture and algorithms for a scalable sending of the outcome of continuous actions in the context of the given target P2P-MMVE system model. At next, Chapter 1 gave a short overview of the conceptual characteristics of the continuous events approach. The continuous events approach builds on the notion that informations about continuous actions are sent only once over the network instead of sending multiple events describing the outcome of the continuous actions over time. Based on the sent informations, the outcome can then be calculated on the peers. The informations about continuous actions are propagated based on continuous events. Continuous events are dedicated event entities for continuous actions that carry the informations and reside within the system over the duration of the continuous actions. Continuous events are executed and managed automatically by the system. Generic function code can be added to the system and is called by continuous events. In the following, Chapter 1 summarized the contributions of this work to the research field of P2P-MMVEs. The continuous events approach enhances the scalability of P2P-MMVEs by propagating the future outcome of continuous actions in an aggregated way. One continuous event is able to describe complex state changes of multiple objects and to apply state changes to a potentially huge number of objects. The use of continuous events is facilitated by given interfaces. Execution and management of continuous events is encapsulated. The approach uses generic

function code that enables the support of a wide variety of continuous action types and use cases. Finally, Chapter 1 presented the organization of this work.

Chapter 2 described the background of this work. At first, Chapter 2 introduced MMVEs and gave an overview of MMVE systems from research and practice. MMVEs originated primarily from military research and initiatives that aimed at providing simulations for military training. Over the years, more and more fields of usage evolved and even several IEEE standards were published. A research community specifically for P2P-MMVEs formed in the 2000s. At next, Chapter 2 identified related work. This work has relations to the fields of interest management, dead reckoning and several prediction techniques for avatar behavior. An overview of appproaches from these fields was given. In addition, the concept of Geocast was described, which might be a potential additional field of usage for the continuous events approach. A summary of own related work was given. After that, Chapter 2 presented the target P2P-MMVE system model of this work. This work originated from a research project that aimed at providing a framework for P2P-MMVEs. The target P2P-MMVE system model includes the following characteristics: Communication via peer-to-peer, use of a superpeer concept by tesselating the virtual environment into zones and assigning peers as superpeers to zones for coordination of system tasks, stateful objects, event-driven propagation of state changes based on the spatial publish subscribe communication model, existence of a MMVE software on all peers. Chapter 2 concluded with an overview of the underlying assumptions and the requirements for this work.

Chapter 3 presented the continuous events approach. In the continuous events approach, continuous actions are represented by continuous events. A continuous event is an explicit event entity for representing continuous actions in the system. A continuous event carries informations that describe the future outcome of a continuous action. It includes informations about the spatial influence of a continuous action over time, its effect on the virtual environment and objects over time, and timing information. Instead of propagating events about every state change caused by a continuous action to other peers, a continuous event is propagated and the future outcome is calculated and applied on these peers. In the following, Chapter 3 discussed varying modeling alternatives for the spatial influence of a continuous action over time. Because there is not one best way to model the spatial influence of all types of continuous actions, the continuous events approach supports spatial influence areas for continuous actions in a generic way. Function code for calculating

certain influence areas over time can be added to the system and continuous events call these functions to get the shape of the spatial influence over time. The calculation of the effect of a continuous action on the virtual environment is supported in the same generic way. Function code can be added that calculates the effect of a continuous event over time and is called by the continuous event. At next, Chapter 3 introduced a formal model for continuous events. Finally, Chapter 3 concluded with a discussion of timing alternatives. Continuous events can be either finite or infinite. Finite continuous events have a clearly defined start and end and are automatically terminated by the system. Infinite continuous event have a clearly defined start but no end. They have to be terminated explicitly. The continuous events approach supports both types. The generic support of the spatial influence and effect of continuous actions and the support of both timing alternatives allows to support a wide variety of use cases.

Chapter 4 presented a system architecture that supports the continuous events approach in the context of the given P2P-MMVE target system model. At first, Chapter 4 gave an overview of the overall design of the system architecture and the integration into the target system model. The system architectures includes architectural components for coordination of distributed communication, managing existing continuous events in the system, executing continuous events on all peers, providing function code on all peers, storing continuous events on all peers, and for registering timers. In addition, functionalities of several other architectural components of the target system are used by the system architecture. In the following, Chapter 4 described the basic continuous event support by the system architecture. The system behavior, algorithms and information flow were explained in detail for the following basic use cases: MMVE start, joining of a new peer, leaving of a peer, start and execution of continuous events, management of existing continuous events, modification and termination of existing continuous events. Finally, Chapter 4 presented and discussed several extensions.

Chapter 5 described an evaluation of the continuous events approach. At first, Chapter 5 discussed the focus of the presented evaluation. Because preliminary load experiments showed that calculation load is not a limiting factor for the continuous events approach, the presented evaluation focuses on the networking aspects of the approach. At next, Chapter 5 explained the methodology of the evaluation. The evaluation was performed via simulations of the approach. Several aspects for an assessment of the simulation results were identified. Two metrics for recording

results, message number and payload size, were derived from the identified aspects. After that, Chapter 5 gave an overview of the Java implementation of a software prototype for simulation of the continuous events approach. The prototype includes basic continuous event support. It is able to simulate the information flow within a zone of a P2P-MMVE according to the target system model based on pre-recorded movement and action data. In the following, Chapter 5 described the process of a typical simulation run, the software and hardware specifications for the simulations, and the general settings of the simulation. The general settings emulate the conditions of a Second Life zone. Simulations were performed for four types of continuous actions with varying characteristics. For each action type, the information flow of a basic P2P-MMVE approach with spatial publish subscribe, a client/server-based approach with dead reckoning, and two alternatives of the continuous events approach were simulated. The two alternatives of the continuous events approach used varying modelings of the spatial influence of the continuous actions: A minimum and a maximum spatial modeling.

Chapter 5 concluded with an assessment and discussion of the simulation results. In comparison to the simulated basic P2P-MMVE approach, the continuous events approach was able to reduce network traffic significantly for all four types of continuous actions. A comparison to the client/server-based approach with dead reckoning showed that the continuous events approach came close, but could not beat the reduction performance of dead reckoning for continuous actions that only or predominantly involve movement. For more complex continuous actions, the continuous events approach had clear advantages. The spatial modeling had a direct implication on the characteristics of the information flow. A minimum modeling led to less overall network traffic but a higher load on the peer that manages existing continuous events. A maximum modeling led to a higher overall network traffic but more continuous events were propagated directly between peers and, as a result, the network load was more distributed over all peers. A comparison of the payload size showed that the payload of continuous events was only slightly larger than the payload of messages for dead reckoning.

6.2. Classification of this Work

The fields of dead reckoning and prediction techniques for avatar behavior are direct related work to this work. An overview of approaches from both fields was given

earlier in Subsection 2.2. In the following, the presented continuous events approach is compared to the general concepts of dead reckoning approaches and approaches with prediction techniques for avatar behavior.

All three types of approaches build on a similar notion. Instead of sending accurate updates for each state change that is caused by a continuous action, the approaches accept a certain grade of inaccuracy in order to enhance the scalability of propagation. Informations describing the outcome are sent and future state changes are exrapolated based on mathematical equations or are predicted based on algorithms. The similarity of the underlying notion becomes apparent when comparing Figure 2.2 on page 30, Figure 2.3 on page 36 and Figure 3.3 on page 55.

However, a comparison of the three figures also shows the differences of the approaches. Dead reckoning clearly aims at optimizing the propagation of movement. Future positions are calculated based on mathematical equations that reside on each user computer. In contrast to the other approaches, dead reckoning typically includes algorithms that regularly check for position deviations and trigger a resending of parameters and converging of positions in case the deviation exceeds a given threshold. Because dead reckoning and its algorithms are tailored specifically towards positions and movement, the approach has advantages for a scalable propagation of continuous actions that only or predominantly involve movement. This claim was backed up by the simulation results presented in this work.

Prediction techniques for avatar behavior can conceptually be classified closer to the continuous events approach because the supported types of continuous actions are more generic than the continuous actions supported by dead reckoning. Prediction techniques for avatar behavior communicate a rough description of future avatar behavior and use function code or artificial intelligence routines to let the copies of user avatars act similar to the actual user avatar. In contrast to the continuous events approach, prediction techniques for avatar behavior typically perform their optimization correlated to a certain avatar object. The continuous events approach, on the other hand, is able to manipulate the state of a large number of objects of varying types. Instead of defining certain code or artificial intelligence routines, the continuous events approach supports a wide variety of use cases by allowing the adding of generic function code to the system.

From a conceptual point of view, the continuous events approach has the highest grade of abstraction. Various types of continuous actions are supported. This might result in a slightly worse performance compared to the other approaches

when confronted with the specific continuous action type that is supported by these approaches. For example, the simulated client/server system with dead reckoning was able to beat the continuous events approach for continuous actions that only or predominantly involve movement. However, the continuous events approach is able to support the widest variety of use cases.

6.3. Fulfillment of Requirements

Section 2.5 identified the following requirements for this work:

1. Scalable propagation of continuous actions
2. Support of explicit use
3. Encapsulation of execution and management processes by the system
4. Handling of peer crashes
5. Handling of peer disconnections
6. Handling of overloaded peers

The performed simulations showed that the continuous events approach was able to reduce the number of overall messages and payload for propagation of continuous actions significantly in comparison to the simulated event-driven P2P-MMVE system with an event propagation via spatial publish subscribe using single events. In addition, the simulations showed that for continuous actions that predominantly involve movement the continuous events approach was able to come at least close in message and payload reduction to the simulated client/server-based MMVE system with dead reckoning. For continuous actions with more complex characteristics involving multiple influences on other objects in addition to movement, the continuous events approach clearly beat the client/server-based system with dead reckoning. The reduction of the overall number of messages and payload is directly correlated to the scalability of the event propagation of a P2P-MMVE system. Therefore, based on the strong reduction rate it can be concluded that the continuous events approach is able to propagate continuous actions in P2P-MMVE systems in a scalable way. *Requirement 1* is fulfilled by the presented approach.

The presented system architecture for continuous event support includes interfaces that allow the explicit start, modification and termination of continuous events by the MMVE software. Algorithms for the start, modification and termination of

continuous events were described in this work. *Requirement 2* is fulfilled by the presented approach.

The system architecture includes architectural components that execute continuous events on the peers (CEE) and manage existing continuous events in the system (CEM). After the explicit start of a continuous event by the MMVE software, the components execute and manage continuous events automatically. Algorithms for execution and management were presented in this work. The propagation of an explicit modification or termination of an existing continuous event by the MMVE software is performed automatically by the system architecture. The CEM administrates receiver lists for continuous events and delivers informations about modifications or termination of a continuous event based on the receiver list of this continuous event to all peers that have a copy of the continuous event. *Requirement 3* is fulfilled by the presented approach.

Subsections 4.3.2, 4.3.3 and 4.3.4 discussed potential solutions for the challenges of peer crashes, peer disconnections and overloaded peers. No specific algorithms were designed because such algorithms are closely related to other framework parts and solutions have to be found at the scope of the overall system architecture, which is beyond the scope of this work. However, potential strategies and solutions for all challenges were at least discussed. At the scope of this work, no better fulfillment of *requirements 4, 5 and 6* can be reached.

6.4. Future Directions

Subsection 2.2.4 of this work observed that the concept of geocast from the research fields of mobile and context-aware computing shows similarities to the spatial publish subscribe communication mechanism that is used for propagation of continuous events by the continuous events approach. The similarities between geocast and spatial publish subscribe might allow to use the concept of continuous events in research fields beyond P2P-MMVEs. A potential direction for future research might be to explore the use of continuous events in mobile computing infrastructures based on geocast. In order to enable the use of continuous events in infrastructures with geocast, the similarities and differences between geocast and spatial publish subscribe have to be researched in more detail. Potentially needed adjustments to the continuous events approach for a use in the context of geocast have to be identified and applied to the presented architectural design and algorithms.

Another potential direction for future research or, at least, for an extension of the continuous events approach might be a spatial propagation using multi-dimensional shapes. The spatial publish subscribe mechanism that is used by the continuous events approach manages interest based on a 2-dimensional spatial projection of the virtual environment. This simplification reduces the burden for calculating intersections between spatial AoE publications and spatial AoI subscriptions and constitutes a trade-off between accuracy of spatial publish subscribe and calculation overhead. Nevertheless, expanding the spatiality towards a third dimension is desirable. MMVEs often include user actions such as flying that result in 3-dimensional positions of user avatars. Such avatars are positioned high above other user avatars and the peers of these users do not have to be provided with updates about each other. In a 2-dimensional projection, these peers might still be provided with updates of each other because they might have the same coordinates on a 2-dimensional plane. In such scenarios, spatial publish subscribe can be performed more accurate with 3-dimensional shapes. However, intersecting and managing 3-dimensional shapes might put a high calculation burden on the peers. Therefore, instead of adding a full third dimension, a potential trade-off might be found in the field of location models for geocast. In [DR03], the authors use 2.5-dimensional shapes that consist of a 2-dimensional base and an altitude value. In order to check if intersections between two shapes exist, the bases have to be intersected and the altitude values have to be compared. The grade of accuracy of 2.5-dimensional shapes for spatial publish subscribe is lower than a full 3-dimensional scheme. However, altitude values of flying users can be considered for spatial publish subscribe with low calculation overhead. Extending the spatiality of the spatial publish subscribe mechanism that is used by the continuous events approach with 2.5-dimensional shapes might be a future extension for the continuous events approach.

Over the years, numerous P2P-MMVE system architectures were proposed. It was shown that the peer-to-peer model is a reasonable alternative to MMVEs based on the client/server model. Nevertheless, at the point of writing of this work the state-of-the-art architecture that is used by commercially successful MMVEs still is typically based on the client/server model. The client/server model is well understood and allows MMVE providers to have a high grade of control over the system. Based on the client/server model, companies from the gaming sector are able to provide MMVEs for several hundered simultaneous users in a scalable way. Assuming in the future the state-of-the-art system architecture will still be client/server-based, exploring the use of the continuous events approach in the context of the client/server

model might also be a direction of future research. The evaluations that were performed and described in this work used one CEM. This CEM could be placed on the server in a client/server-based system. Each client could run a CEE, similar to the regular peers in the simulations. Under the assumption that the client/server-based system includes a spatial publish subscribe network service and a service for addressing messages directly to certain clients, the continuous events approach could be adjusted for use in a client/server-based system. Because one CEM was used, the presented simulation results not only indicate a high reduction potential by the continuous events approach in a P2P-MMVE. In addition, the results hint at a similar potential in the context of client/server-based systems. The design of a modified system architecture for continuous event support in a client/server-based system and the exploration of the potential of the continuous events approach in the context of the client/server model might be an interesting direction for future research.

The simulations further showed that dead reckoning has an advantage over the continuous events approach for the propagation of continuous actions that only involve object movement. In contrast, the continuous events approach is able to support more continuous action types and has an advantage over dead reckoning for the propagation of complex continuous actions. A potential field of future research might be to explore the mutual influence of the approaches when used in conjunction in the same MMVE system. In case the approaches can be used simultaneously without a negative mutual influence, dead reckoning could be used for the propagation of pure movement and the continuous events approach for other types of continuous actions. Such a solution has the potential to combine the advantages of both approaches. Exploring the mutual influence between dead reckoning and the continuous events approach might be another promising direction for future research.

A. Simulation Results

	Messages Sent		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	639291
Client/Server with Dead Reckoning	13842	0	300
P2P SPS with CE Min	0	11809	4366
P2P SPS with CE Max	0	1853	28086
	Messages Received		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P with SPS)	0	0	58995683
Client/Server with Dead Reckoning	1716840	0	36960
P2P SPS with CE Min	0	1609378	493552
P2P SPS with CE Max	0	252524	3133808
	Payload Sent (Byte)		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	639291
Client/Server with Dead Reckoning	300	0	13842
P2P SPS with CE Min	0	2333	13842
P2P SPS with CE Max	0	14193	15746
	Payload Received (Byte)		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	58995683
Client/Server with Dead Reckoning	36960	0	1716840
P2P SPS with CE Min	0	215818	1887112
P2P SPS with CE Max	0	1235778	2150554
	Payload Comparison (Byte)		
	Messages	Payload	Payload per Message
Dead Reckoning	14142	1753800	124.0135766
CE Min	14142	1928092	136.3380003
CE Min Registration	2033	174838	86
CE Max	16046	2191534	136.5782126
CE Max Registration	13893	1194798	86

Table A.1.: Recorded simulation results for continuous action type 1

	Messages Sent		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	642506
Client/Server with Dead Reckoning	17057	0	588
P2P SPS with CE Min	0	11949	4366
P2P SPS with CE Max	0	1903	28296
	Messages Received		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	59328892
Client/Server with Dead Reckoning	1971054	0	66797
P2P SPS with CE Min	0	1628376	493552
P2P SPS with CE Max	0	259337	3157176
	Payload Sent (Byte)		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	642506
Client/Server with Dead Reckoning	588	0	17057
P2P SPS with CE Min	0	2333	13982
P2P SPS with CE Max	0	14298	15901
	Payload Received (Byte)		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	59328892
Client/Server with Dead Reckoning	66797	0	1971054
P2P SPS with CE Min	0	215818	1906110
P2P SPS with CE Max	0	1244808	2171705
	Payload Comparison (Byte)		
	Messages	Payload	Payload per Message
Dead Reckoning	14142	1753800	124.0135766
CE Min	14282	1947090	136.3317463
CE Min Registration	2033	174838	86
CE Max	16201	2212685	136.5770631
CE Max Registration	13998	1203828	86

Table A.2.: Recorded simulation results for continuous action type 2

	Messages Sent		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	1467224
Client/Server with Dead Reckoning	838910	0	34656
P2P SPS with CE Min	0	12830	6030
P2P SPS with CE Max	0	1862	31662
	Messages Received		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	144506858
Client/Server with Dead Reckoning	81115107	0	3583109
P2P SPS with CE Min	0	1748354	678750
P2P SPS with CE Max	0	253749	3531873
	Payload Sent (Byte)		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	1467224
Client/Server with Dead Reckoning	34656	0	838910
P2P SPS with CE Min	0	3165	15695
P2P SPS with CE Max	0	15981	17543
	Payload Received (Byte)		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	144506858
Client/Server with Dead Reckoning	3583109	0	81115107
P2P SPS with CE Min	0	287370	2139734
P2P SPS with CE Max	0	1389546	2396076
	Payload Comparison (Byte)		
	Messages	Payload	Payload per Message
Dead Reckoning	14142	1753800	124.0135766
CE Min	15995	2180714	136.3372304
CE Min Registration	2865	246390	86
CE Max	17843	2437056	136.58331
CE Max Registration	15681	1348566	86

Table A.3.: Recorded simulation results for continuous action type 3

	Messages Sent		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	29886020
Client/Server with Dead Reckoning	16906830	0	693120
P2P SPS with CE Min	0	22130	14868
P2P SPS with CE Max	0	0	59700
	Messages Received		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	2943966445
Client/Server with Dead Reckoning	1637564130	0	71597832
P2P SPS with CE Min	0	3017607	1657848
P2P SPS with CE Max	0	0	6634200
	Payload Sent (Byte)		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	29886020
Client/Server with Dead Reckoning	693120	0	16906830
P2P SPS with CE Min	0	7584	29414
P2P SPS with CE Max	0	30000	29700
	Payload Received (Byte)		
	Server	Managing Peer	Clients / Regular Peers
No Optimization (P2P SPS)	0	0	2943966445
Client/Server with Dead Reckoning	71597832	0	1637564130
P2P SPS with CE Min	0	667224	4008231
P2P SPS with CE Max	0	2595000	4039200
	Payload Comparison (Byte)		
	Messages	Payload	Payload per Message
Dead Reckoning	51595	6345982	122.9960655
CE Min	29714	4049031	136.2667766
CE Min Registration	7284	626424	86
CE Max	30000	4080000	136
CE Max Registration	29700	2554200	86

Table A.4.: Recorded simulation results for continuous action type 4

B. Load Experiment

Property	Value
Operating System	Windows 7 Professional (Service Pack 1), 64 Bit
Java Virtual Machine	Version 7, Update 7
CPU	Intel Core2Duo, E7200, 2.53GHz
Memory	4 GB

Table B.1.: Load experiment: Software and hardware specifications

	Action Type 1	Action Type 2	Action Type 3	Action Type 4
Run 1	6250	6119	5877	5554
Run 2	6077	5725	5801	5703
Run 3	5928	5800	5754	5765
Run 4	6174	6275	5809	5530
Run 5	5943	6535	5623	5807
Run 6	6190	6268	5794	5798
Run 7	6203	5782	5620	5837
Run 8	6181	5740	5783	5905
Run 9	6036	6148	5794	5765
Run 10	5969	5891	5805	5572
Average	6095.1	6028.3	5766	5723.6

Table B.2.: Load experiment: Number of calculated steps for CE Min AoE in 100 ms

	Action Type 1	Action Type 2	Action Type 3	Action Type 4
Run 1	5835	5486	5369	5445
Run 2	6151	5846	5311	5181
Run 3	5849	5824	5514	5219
Run 4	5820	5566	5494	5716
Run 5	5918	5627	5690	5724
Run 6	6017	5641	5633	5520
Run 7	5965	5996	5670	5433
Run 8	6182	5606	5557	5278
Run 9	5735	5719	5561	5585
Run 10	6008	5746	5458	5336
Average	5948	5705.7	5525.7	5443.7

Table B.3.: Load experiment: Number of calculated steps for CE Max AoE in 100 ms

Bibliography

[ARB90] John M. Airey, John H. Rohlf, and Frederick P. Brooks, Jr. Towards image realism with interactive update rates in complex virtual building environments. In *Proceedings of the 1990 Symposium on Interactive 3D Graphics*, pages 41–50, March 1990.

[Aro] Jesse Aronson. Dead reckoning: Latency hiding for networked games. Online article, September 1997. Accessed: 13.09.2011,12:16. http://www.gamasutra.com/view/feature/3230/dead_reckoning_latency_hiding_for_.php.

[ASdO06] Dewan Tanvir Ahmed, Shervin Shirmohammadi, and Jauvane C. de Oliveira. A novel method for supporting massively multi-user virtual environments. In *Proceedings of the IEEE International Workshop on Haptic Audio Visual Environments and their Applications (HAVE)*, pages 72–77, November 2006.

[ATS04] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR)*, 36(4):335–371, December 2004.

[BC85] Eric J. Berglund and David R. Cheriton. Amaze: A multiplayer computer game. *IEEE Software*, 2:30–39, May 1985.

[BCL+04] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. The effects of loss and latency on user performance in Unreal Tournament 2003. In *Proceedings of the 3rd Workshop on Network and System Support for Games (NetGames)*, pages 144–151, August 2004.

[BD05] Christian Becker and Frank Dürr. On location models for Ubiquitous Computing. *Personal Ubiquitous Computing*, 9(1):20–31, January 2005.

[BDL+08] Ashwin Bharambe, John R. Douceur, Jacob R. Lorch, Thomas Moscibroda, Jeffrey Pang, Srinivasan Seshan, and Xinyu Zhuang. Donny-

brook: Enabling large-scale, high-speed, peer-to-peer games. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, pages 389–400, August 2008.

[BF93] Steve Benford and Lennart Fahlén. A spatial model of interaction in large virtual environments. In *Proceedings of the Third European Conference on Computer-Supported Cooperative Work (ECSCW)*, pages 109–124, September 1993.

[BGRP01] Steve Benford, Chris Greenhalgh, Tom Rodden, and James Pycock. Collaborative virtual environments. *Communications of the ACM*, 44(7):79–85, July 2001.

[BHS+08] Jean Botev, Alexander Höhfeld, Hermann Schloss, Ingo Scholtes, and Markus Esch. The HyperVerse - concepts for a federated and torrent-based 3D web. In *Proceedings of the 1st International Workshop on Massively Multiuser Virtual Environments (MMVE)*, pages 34–39, March 2008.

[BKR07] Olivier Beaumont, Anne-Marie Kermarrec, and Étienne Rivière. Peer to peer multidimensional overlays: Approximating complex structures. In *Proceedings of the 11th International Conference on Principles of Distributed Systems*, pages 315–328, December 2007.

[BKV06] Jean-Sébastien Boulanger, Jörg Kienzle, and Clark Verbrugge. Comparing interest management algorithms for massively multiplayer games. In *Proceedings of the 5th Workshop on Network and System Support for Games (NetGames)*, page 6, October 2006.

[Blia] Blizzard Entertainment, Inc. Official website of World of Warcraft. Online. Accessed: 18.11.2012,11:43. http://www.worldofwarcraft.com.

[Blib] Blizzard Entertainment, Inc. World of Warcraft subscriber base reaches 12 million worldwide. Online press release, October 2010. Accessed: 19.10.2012,11:24. http://us.blizzard.com/en-us/company/press/pressreleases.html?id=2847881.

[BMJ+98] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 85–97, October 1998.

[BPS06] Ashwin Bharambe, Jeffrey Pang, and Srinivasan Seshan. Colyseus: A distributed architecture for online multiplayer games. In *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI)*, pages 155–168, May 2006.

[BRS02] Ashwin Bharambe, Sanjay Rao, and Srinivasan Seshan. Mercury: A scalable publish-subscribe system for internet games. In *Proceedings of the 1st Workshop on Network and System Support for Games (NetGames)*, pages 3–9, 2002.

[CC05] Ling Chen and Gencai Chen. A fuzzy dead reckoning algorithm for distributed interactive applications. In *Proceedings of the Second international Conference on Fuzzy Systems and Knowledge Discovery*, pages 961–971, August 2005.

[CET99] Tolga K. Capin, Joaqium Esmerado, and Daniel Thalmann. A dead-reckoning technique for streaming virtual human animation. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(3):411–414, April 1999.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

[CYB+07] Luther Chan, James Yong, Jiaqiang Bai, Ben Leong, and Raymond Tan. Hydra: A massively-multiplayer peer-to-peer architecture for the game developer. In *Proceedings of the 6th Workshop on Network and System Support for Games (NetGames)*, pages 37–42, 2007.

[DG03] Thomas P. Duncan and Denis Gracanin. Pre-reckoning algorithm for distributed virtual environments. In *Proceedings of the 2003 Winter Simulation Conference*, pages 1086–1093, December 2003.

[dOG02] Jauvane C. de Oliveira and Nicolas D. Georganas. VELVET: An adaptive hybrid architecture for very large virtual environments. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 2491–2495, May 2002.

[DR03] Frank Dürr and Kurt Rothermel. On a location model for fine-grained geocast. In *Proceedings of the 5th International Conference on Ubiquitous Computing (UbiComp)*, pages 18–35, October 2003.

[DTHK05] Scott Douglas, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera. Enabling massively multi-player online gaming applica-

tions on a P2P architecture. In *Proceedings of the IEEE International Conference on Information and Automation*, pages 7–12, December 2005.

[ERM05] Abdennour El Rhalibi and Madjid Merabti. Agents-based modeling for a peer-to-peer MMOG architecture. *Computers in Entertainment*, 3(2):3–3, April 2005.

[FRP+08] Davide Frey, Jérôme Royan, Romain Piegay, Anne marie Kermarrec, Fabrice Le Fessant, and Emmanuelle Anceaume. Solipsis: A decentralized architecture for virtual environments. In *Proceedings of the 1st International Workshop on Massively Multiuser Virtual Environments (MMVE)*, pages 29–33, March 2008.

[Fuj98] Richard M. Fujimoto. Time management in the High Level Architecture. *Simulation*, 71(6):388–400, December 1998.

[GB95] Chris Greenhalgh and Steven Benford. MASSIVE: A collaborative virtual environment for teleconferencing. *ACM Transactions on Computer-Human Interaction (TOCHI) - Special Issue on Virtual Reality Software and Technology*, 2(3):239–261, September 1995.

[GLZ05] Chris GauthierDickey, Virginia Lo, and Daniel Zappala. Using n-trees for scalable event ordering in peer-to-peer games. In *Proceedings of the 15th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 87–92, June 2005.

[Gre97] Chris Greenhalgh. *Large Scale Collaborative Virtual Environments*. PhD thesis, University of Nottingham, October 1997.

[HBH06] Thorsten Hampel, Thomas Bopp, and Robert Hinn. A peer-to-peer architecture for massive multiplayer online games. In *Proceedings of the 5th Workshop on Network and System Support for Games (NetGames)*, page 48, 2006.

[HCC06] Shun-Yun Hu, Jui-Fa Chen, and Tsu-Han Chen. VON: A scalable peer-to-peer network for virtual environments. *IEEE Network*, 20(4):22–31, July-August 2006.

[HL04] Shun-Yun Hu and Guan-Ming Liao. Scalable peer-to-peer networked virtual environment. In *Proceedings of the 3rd Workshop on Network*

and System Support for Games (NetGames), pages 129–133, August 2004.

[HLL00] Seunghyun Han, Mingyu Lim, and Dongman Lee. Scalable interest management using interest group based filtering for large networked virtual environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 103–108, October 2000.

[HLLH08] Seunghyun Han, Mingyu Lim, Dongman Lee, and Soon J. Hyun. A scalable interest management scheme for distributed virtual environments. *Comput. Animat. Virtual Worlds*, 19(2):129 – 149, May 2008.

[HSS+12] Florian Heger, Gregor Schiele, Richard Süselbeck, Laura Itzel, and Christian Becker. Scalability in peer-to-peer-based MMVEs: The continuous events approach. In *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC) 2012*, pages 629–633, January 2012.

[HSSB09] Florian Heger, Gregor Schiele, Richard Süselbeck, and Christian Becker. Towards an interest management scheme for peer-based virtual environments. In *Proceedings of the 1st International Workshop on Concepts of Massively Multiuser Virtual Environments (COMMVE)*, March 2009.

[Hu09] Shun-Yun Hu. Spatial publish subscribe. In *Proceedings of the 2nd International Workshop on Massively Multiuser Virtual Environments (MMVE)*, March 2009.

[HWB+10] Shun-Yun Hu, Chuan Wu, Eliya Buyukkaya, Chien-Hao Chien, Tzu-Hao Lin, Maha Abdallah, Jehn-Ruey Jiang, and Kuan-Ta Chen. A spatial publish subscribe overlay for massively multiuser virtual environments. In *Proceedings of the 2010 International Conference On Electronics and Information Engineering (ICEIE)*, pages 314–318, August 2010.

[iee93] IEEE Std 1278-1993. IEEE Standard for Information Technology - Protocols for Distributed Interactive Simulations Applications–Entity Information and Interaction, 1993.

[iee95a] IEEE Std 1278.1-1995. IEEE Standard for Distributed Interactive Simulation - Application Protocols, 1995.

[iee95b] IEEE Std 1278.2-1995. IEEE Standard for Distributed Interactive Simulation - Communication Services and Profiles, 1995.

[iee96] IEEE Std 1278.3-1996. IEEE Recommended Practice for Distributed Interactive Simulation - Exercise Management and Feedback, 1996.

[iee97] IEEE Std 1278.4-1997. IEEE Trial-Use Recommended Practice for Distributed Interactive Simulation - Verification, Validation, and Accreditation, 1997.

[iee98] IEEE Std 1278.1a-1998. IEEE Standard for Distributed Interactive Simulation - Application Protocols, 1998.

[iee03] IEEE Std 1516.3-2003. IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP), 2003.

[iee07] IEEE Std 1516.4-2007. IEEE Recommended Practice for Verification, Validation, and Accreditation of a Federation - an Overlay to the High Level Architecture Federation Development and Execution Process, 2007.

[iee10a] IEEE Std 1516-2010. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules, 2010.

[iee10b] IEEE Std 1516.1-2010. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification, 2010.

[iee10c] IEEE Std 1516.2-2010. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template (OMT) Specification, 2010.

[IHK04] Takuji Iimura, Hiroaki Hazeyama, and Youki Kadobayashi. Zoned federation of game servers: A peer-to-peer approach to scalable multiplayer online games. In *Proceedings of the 3rd Workshop on Network and System Support for Games (NetGames)*, pages 116–120, August 2004.

[ISST06] Shin Ito, Hajime Saito, Hiroki Sogawa, and Yoshito Tobe. A propagation of virtual space information using a peer-to-peer architecture for massively multiplayer online games. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW)*, page 44, July 2006.

[JM96] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[KLXH04] Björn Knutsson, Honghui Lu, Wei Xu, and Bryan Hopkins. Peer-to-peer support for massively multiplayer games. In *Proceedings of the 23rd Conference of the IEEE Communications Society (INFOCOM)*, March 2004.

[KMA02] Yoshihiro Kawahara, Hiroyuki Morikawa, and Tomonori Aoyama. A peer-to-peer message exchange scheme for large scale networked virtual environments. In *Proceedings of the 8th International Conference on Communication Systems (ICCS)*, pages 957–961, April 2002.

[KS02] Joaquín Keller and Gwendal Simon. Toward a peer-to-peer shared virtual reality. In *Proceedings of IEEE Workshop on Resource Sharing in Massively Distributed Systems (RESH)*, pages 595–601, July 2002.

[KS03] Joaquín Keller and Gwendal Simon. Solipsis: A massively multi-participant virtual world. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 262–268, June 2003.

[Kul09] Santosh Kulkarni. Badumna network suite: A decentralized network engine for massively multiplayer online applications. In *Proceedings of the IEEE Ninth International Conference on Peer-to-Peer Computing (P2P)*, pages 178–183, September 2009.

[LBW97] Kuo-Chi Lin, Jesse L. Blair, and John M. Woodyard. Study on dead-reckoning translation in High Level Architecture. *Simulation*, 69(2):103–109, August 1997.

[LCP+05] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7(2):72–93, 2005.

[LCTC00] Bu-Sung Lee, Wentong Cai, Stephen J. Turner, and L. Chen. Adaptive dead reckoning algorithms for Distributed Interactive Simulation. *International Journal of Simulation Systems, Science and Technology*, 1(1):21–34, December 2000.

[Lina] Linden Research, Inc. Official website of Second Life. Online. Accessed: 18.11.2012,16:12. http://www.secondlife.com.

[Linb] Linden Research, Inc. Second Life Wiki. Online. Accessed: 29.01.2013,11:29. http://wiki.secondlife.com/wiki/Main_Page.

[Linc] Linden Research, Inc. Second Life Wiki: Voluntary movement speeds. Online. Accessed: 22.05.2012,15:26. http://wiki.secondlife.com/wiki/Voluntary/_Movement/_Speeds.

[LLI+05] Jinwon Lee, Hyonik Lee, Sunghwan Ihm, Tcaesvk Gim, and Junehwa Song. APOLO: Ad-hoc peer-to-peer overlay network for massively multi-player online games. Technical report, KAIST, December 2005.

[LMS] Paul J. Leach, Michael Mealling, and Rich Salz. Request for Comments (RFC): 4122 - A Universally Unique IDentifier (UUID) URN Namespace. Online, July 2005. Accessed: 29.07.2012,16:11. http://www.ietf.org/rfc/rfc4122.txt.

[MBZ+95] Michael R. Macedonia, Donald P. Brutzman, Michael J. Zyda, David R. Pratt, Paul T. Barham, John Falby, and John Locke. NPSNET: A multi-player 3D virtual environment over the internet. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics (I3D)*, pages 93–94, April 1995.

[MGBY99] Yohai Makbily, Craig Gotsman, and Reuven Bar-Yehuda. Geometric algorithms for message filtering in decentralized virtual environments. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics (I3D)*, pages 39–46, May 1999.

[MMOnD06] Pedro Morillo, W. Moncho, Juan M. Orduña, and José Duato. Providing full awareness to distributed virtual environments based on peer-to-peer architectures. In *Proceedings of the 24th International Conference on Advances in Computer Graphics (CGI)*, pages 336–347, June 2006.

[MS97] Katherine L. Morse and Jeffrey S. Steinman. Data distribution management in the HLA: Multidimensional regions and physically correct filtering. In *Proceedings of the 1997 Spring Simulation Interoperability Workshop*, pages 343–352, March 1997.

[MT95] Duncan C. Miller and Jack A. Thorpe. SIMNET: The advent of sim-

ulator networking. *Proceedings of the IEEE*, 83(8):1114–1123, August 1995.

[MZP+95] Michael R. Macedomia, Michael J. Zyda, David R. Pratt, Donald P. Brutzman, and Paul T. Barham. Exploiting reality with multicast groups: A network architecture for large-scale virtual environments. In *Proceedings of the 1995 IEEE Virtual Reality Annual International Symposium*, pages 2–10, March 1995.

[Ope] OpenSimulator Project. Official website of the OpenSimulator project. Online. Accessed: 21.10.2012,10:49. http://www.opensimulator.org.

[Oraa] Oracle Corporation. Java Platform Standard Edition 7 Online Documentation, class java.lang.Integer. Online. Accessed: 30.10.2012. http://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html.

[Orab] Oracle Corporation. Java Platform Standard Edition 7 Online Documentation, class java.lang.Long. Online. Accessed: 30.10.2012. http://docs.oracle.com/javase/7/docs/api/java/lang/Long.html.

[Orac] Oracle Corporation. Java Platform Standard Edition 7 Online Documentation, class java.lang.String. Online. Accessed: 30.10.2012. http://docs.oracle.com/javase/7/docs/api/java/lang/String.html.

[Orad] Oracle Corporation. Java Platform Standard Edition 7 Online Documentation, class java.util.Calendar. Online. Accessed: 30.10.2012. http://docs.oracle.com/javase/7/docs/api/java/util/Calendar.html.

[Orae] Oracle Corporation. Java Platform Standard Edition 7 Online Documentation, class java.util.Hashtable. Online. Accessed: 30.10.2012. http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html.

[Oraf] Oracle Corporation. Java Platform Standard Edition 7 Online Documentation, class java.util.UUID. Online. Accessed: 30.10.2012. http://docs.oracle.com/javase/7/docs/api/java/util/UUID.html.

[Orag] Oracle Corporation. Java Platform Standard Edition 7 Online Documentation, interface java.util.concurrent.ScheduledExecutorService. Online. Accessed: 30.10.2012. http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ScheduledExecutorService.html.

[PaP] Peers@Play Project. Official Website of the Peers@Play Project. Online. Accessed: 03.02.2013,12:50. http://www.peers-at-play.org/.

[PUL07] Jeffrey Pang, Frank Uyeda, and Jacob R. Lorch. Scaling peer-to-peer games in low-bandwidth environments. In *Proceedings of the 6th International Workshop on Peer-to-Peer Systems (IPTPS)*, February 2007.

[SA04] Anthony Steed and Cameron Angus. Frontier sets: A partitioning scheme to enable scalable virtual environments. In *Proceedings of the Eurographics 2004*, 2004.

[SA05] Anthony Steed and Cameron Angus. Supporting scalable peer to peer virtual environments using frontier sets. In *Proceedings of the 2005 IEEE Conference on Virtual Reality (VR)*, pages 27–34, March 2005.

[SC94] Sandeep K. Singhal and David R. Cheriton. Using a position history-based protocol for distributed object visualization. Technical report, Stanford University, 1994.

[SC96] Sandeep K. Singhal and David R. Cheriton. Using projection aggregations to support scalability in distributed simulation. In *Proceedings of the 16th International Conference on Distributed Computing Systems (ICDCS)*, pages 196–206, May 1996.

[Sca] Scalify Pty Ltd. Badumna network suite online documentation, section 2.3, dead reckoning. Online. Accessed: 20.10.2012,15:48. http://www.scalify.com/documentation/Manual/gettingstarted.html.

[SFC00] Dilza Szwarcman, Bruno Feijó, and Mônica Costa. A framework for networked reactive characters. In *Proceedings of the XIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, pages 203–210, October 2000.

[SHWL09] Gregor Schiele, Shun-Yun Hu, Daniel Weiskopf, and Ben Leong. Challenges in designing massively multiuser virtual environments: Experiences from MMVE 2008. *Special Issue International Journal of Advanced Media and Communication*, 2009.

[SKH02] Jouni Smed, Timo Kaukoranta, and Harri Hakonen. Aspects of networking in multiplayer computer games. *The Electronic Library*, 20(2):87–97, 2002.

[SSGB99] Jianping Shi, Thomas J. Smith, John P. Granieri, and Norman I. Badler. Smart avatars in JackMOO. In *Proceedings of the 1999 IEEE Conference on Virtual Reality (VR)*, pages 156–163, March 1999.

[SSSB09] Richard Süselbeck, Gregor Schiele, Sebastian Seitz, and Christian Becker. Adaptive update propagation for low-latency massively multi-user virtual environments. In *Proceedings of the 18th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–6, August 2009.

[SZ08] Anthony Steed and Bingshu Zhu. An implementation of a first-person game on a hybrid network. In *Proceedings of the 1st International Workshop on Massively Multiuser Virtual Environments (MMVE)*, pages 24–28, March 2008.

[Ter] Daniel Terdiman. Second Life: Don't worry, we can scale. Online article, June 2006. Accessed: 27.01.2013,15:37. http://news.cnet.com/2100-1043_3-6080186.html.

[TS91] Seth J. Teller and Carlo H. Séquin. Visibility preprocessing for interactive walkthroughs. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 61–70, 1991.

[WW94] Annette L. Wilson and Richard M. Weatherly. The aggregate level simulation protocol: An evolving system. In *Proceedings of the 26th Winter Simulation Conference (WSC)*, pages 781–787, December 1994.

[YHK11] Amir Yahyavi, Kévin Huguenin, and Bettina Kemme. AntReckoning: Dead reckoning using interest modeling by pheromones. In *Proceedings of the 10th Workshop on Network and System Support for Games (NetGames)*, pages 1–6, October 2011.

[YMYI05] Shinya Yamamoto, Yoshihiro Murata, Keiichi Yasumoto, and Minoru Ito. A distributed event delivery method with load balancing for MMORPG. In *Proceedings of the 4th Workshop on Network and System Support for Games (NetGames)*, pages 1–8, October 2005.

[YV05] Anthony Yu and Son T. Vuong. MOPAR: A mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games. In *Proceedings of the 15th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 99–104, June 2005.

Curriculum Vitae

Seit Februar 2008	Manager Information Systems and IT Services an der Mannheim Business School gGmbH
März 2009 - Februar 2011	Akademischer Mitarbeiter am Lehrstuhl für Wirtschaftsinformatik II, Universität Mannheim, Lehrstuhlinhaber: Prof. Dr. Christian Becker
September 2002 - Mai 2008	Studium in Wirtschaftsinformatik (Diplom) an der Universität Mannheim
Juni 2001	Abitur am Gymnasium am Kaiserdom, Speyer