Exploitation of Event-Semantics for Distributed Publish/Subscribe Systems in Massively Multiuser Virtual Environments

Thomas Fischer, Michael Daum, Florian Irmert, Christoph Neumann, Richard Lenz
Friedrich-Alexander University of Erlangen-Nuremberg
Institute of Computer Science 6 (Data Management)
Martensstraße 3
D-91058 Erlangen
{thomas.fischer, michael.daum, florian.irmert, christoph.neumann, richard.lenz}
@cs.fau.de

ABSTRACT

Triggered by the fast evolving technical capabilities for implementing distributed global scale applications, online games have grown to a huge industry in recent years. Particularly, *Massive Multiuser Virtual Environment*s (MMVEs), which allow for simultaneous activity of thousands of players in a virtual world, have been tremendously successful. Current architectures, however, use centralized approaches, which obviously do not scale beyond a certain point. Distributed event-based systems are a promising approach to reach both, performing and scalable architectures. The potential of this approach can only be fully exploited if event semantics is used to optimize event handling. Existing approaches actually do this to some degree, but typically in a very application specific manner. There is no generally applicable framework for classifying events according to their relevant semantic properties. In this paper, we propose a generally applicable classification of events as a first step on the way to flexibly adaptable generic event management systems. We exemplify the relevance of our semantic properties by classifying typical events in an existing MMVE. We discuss existing optimization strategies based on our semantic classification and outline a corresponding architecture.

1. INTRODUCTION

Computer games have become part of our social culture and grown to an important branch for the computer industry. The most growing market in this area are online games [1]. But this promising market states some unique challenges for game designers and researchers.

MMVEs define a distributed virtual world shared by thousands of participants, each represented by an avatar, who compete and cooperate in one enormous persistent world. This world may be a game world as in *Massive Multiplayer Online Game*s (MMOGs), a large scale simulation or a virtual world like Second Life. The design of such a living virtual world requires armies of artists and the software backing a world of such enormous dimensions must satisfy several requirements, e.g.:

Consistency The virtual world of the MMVE must be consistent for all participants. Every event that happens inside this world has to be recognizable for all affected avatars.

Availability The virtual world of a MMVE should be available 24 hours a day, 7 days a week, because users expect a high availability for the fee they pay. Therefore, down-time and login problems may decrease or even inhibit the success of the project.

Persistence In MMVEs, an environment that preserves the action of its users has to be provided [2].

Interactivity Interactivity is a crucial factor of success for MMVEs because it provides the desired experience. Depending on the genre, avatars communicate with each other, e.g. to give tactical orders or just for social reasons.

Security With millions of users the likelihood of customers trying to cheat increases.

In contrast to conventional computer games, MMVEs must be able to maintain high quality regarding these requirements even if the user-base grows beyond any predictions. Therefore, scalability of an MMVEs architecture is the crucial requirement. This states the challenge of MMVE architectures: Preserve scalability, whilst all other requirements are met to the desired quality standard.

Current industry strength MMVE architectures favour a client/server architectural style, mostly motivated by the fear of cheats and the easy maintainability. Skibinsky showed in [3] that this type of architecture has "high operational cost, capable of serving in the low thousands of users in the same world and having scalability limits for future growth". To cope with more and more players, tremendous effort is invested by the usage of grid approaches like in Linden Lab's *Second Life* [1] or the deployment of large hierar-

[1] `http://secondlife.com`

http://doi.acm.org/10.1145/1866480.1866494

chical clusters like for the operation of CCP's *Eve Online*[2]. These architectures serve currently a confirmed maximum of about 47 thousand users (Eve Online) in one persistent virtual world [4]. They are only capable of serving such an amount of users as long as their avatars are broadly distributed in the virtual world. When avatars start to flock, the servers' load reaches critical levels very fast. This problem is well known amongst users as the "crowding" problem. As a consequence, nearly all currently available MMVEs design their virtual worlds in a way that encourages participants to distribute broadly over the whole world and try to avoid an avatar concentration in one region.

Recent games have shown that game design and client/server architectures alone do not suffice to solve the challenge of scalability. Distributed systems like specialized *Peer-to-peer* (P2P) systems are a promising approach of handling the snowballing number of players. But even in such systems, the main challenge in handling player concentration in one region is the enormous amount of messages (*events*) that have to be delivered to every single player.

Existing classical multiplayer game architectures often use a broadcast mechanism to keep the distributed world state consistent. Ideally, all events are securely delivered and processed in the same order nearly at the same time on all clients. In a distributed environment, this requires an effort of $O(n^2)$, n being the participating clients of the world. It is obvious that architectures do not scale well beyond a certain amount of clients. Multiplayer games like Quake 3 Arena[3] have shown that without any optimizations, the limit is reached by about 64 clients, depending on the required update rate of the game. Enabling scalability beyond that limit states a challenge. Many recent event-dissemination architectures for MMVEs [5, 6, 7, 8, 9] optimize the event dissemination based on one basic idea: Exploitation of event semantics to reduce message exchange between clients. But to our knowledge, all existing approaches address only one or two aspects of the semantics and propose an optimized event-dissemination architecture for that specialized aspect.

In this paper, we provide the first steps towards a generalized approach of utilizing a comprehensive set of different *semantic* properties of events that may arise in an MMVE to reduce the communication overhead between the clients. This paper is organized as follows: In Section 2, we present our semantic classification schema that introduces dimensions along which all possible events may be classified with respect to their optimization potential. Events that belong to the same class have similar requirements regarding their dissemination. In Section 3, we analyze recent research in context of the optimization used for event dissemination. Based on the classification schema and the optimization strategies we describe an architecture that is build on top of a conventional overlay network and controls an optimized distribution of all events in Section 4. In Section 5, we discuss our proposed approach and provide a roadmap for its realization as well as the identification of the main challenges for future work.

2. EVENT SEMANTICS

Events used in MMVEs feature many characteristics like for example a spatial context in which the event is valid that may be exploited to develop specialized architectures for their delivery to the clients. The decision which optimization suites best for a certain event type is a non-trivial challenge, whose solution in our approach needs a solid model of the events semantics. But first, the basic characteristics of event types in MMVEs have to be discussed. When talking about event semantics, we have to distinguish between event types and events, which means for example "item pickup" is an event type and "player x picks up flower y at position z" is an corresponding event.

Events occur between the world and an avatar or between two avatars. The world in a distributed scenario is built by the clients. There is no central server. Therefore, the responsibility for the entities in the world is divided amongst the clients following a certain algorithm. For example, the world is divided in cells and one client is chosen to act as a server for this cell. In a distributed MMVE, all events occur between the clients, which we call nodes in the remainder of the paper, due to the fact that every node may act as a client or a server depending on the event type discussed.

Another relevant aspect to consider is the frequency of one event type. Position updates for example have a high frequency (around 30 events per second), whilst the pickup of an item is a unique event which only occurs from time to time. There are also bursty events, like the usage of a weapon, where each bullet is represented by an event.

Existing approaches concentrate on the improvement of event types which occur with high frequency without considering all possible semantics of event types required by MMVEs. We strive for a more generalized approach which incorporates all relevant event types and provide the best possible dissemination for each special type.

Such a generalized approach requires a fundamental event model, which defines a formal description of optimizable event types. Based on this formal description, different optimization strategies may be applied to the event dissemination. In this section, we propose an initial identification of event classes which may be optimized in different ways regarding their delivery to the recipient(s) based on the application's requirements. We strive for a multidimensional classification schema which models all aspects of performance relevant event semantics. First, we introduce the dimensions we found based on recent research. Afterwards we exemplify the proposed dimensions by the classification of typical event types in an existing multiplayer game.

2.1 Dimensions

The semantics of each event, which is exchanged between clients of a virtual world, may be analyzed and a definition e.g. of the address, the priority, the relationship to other events or the context in which this event is valid can be deduced. In order to classify the event types properly, we propose a multidimensional classification with orthogonal dimensions, as outlined in [10], to model independent aspects of the event semantics. We define disjoint characteristic classes for each dimension. To classify an event type, it has to be assigned to one class along each dimension. Based on this scheme, the class of an event type is defined as the sum of the characteristics along each dimension. The power of such a multidimensional class space enables optimization of each event type along each dimension with a different strategy in order to gain a better overall optimization foot-

[2] `http://www.eveonline.com/`

[3] `http://www.idsoftware.com/games/quake/quake3-arena/`

print of the system, than if a fixed strategy is used for the whole system. The dimensions proposed below, may not exhaustively address all possible semantic properties events can adopt, but those relevant to existing performance optimization strategies, as discussed in Section 3. The following initial dimensions are found and described with a set of initial classes:

2.1.1 Context

Each event in an MMVE has a certain context in which it is relevant or valid. This context may be e.g. spatial, social or defined by certain metrics [6]. In general, the context of an event in an MMVE reduces its recipients to a certain subset. Most optimization algorithms in this field may be summarized under the topic *Area of Interest* (AoI) management [11]. For the context dimension, following initial classes are defined:

single-target Obviously, an event with only one recipient may be delivered directly, without, e.g, the need of a multicast tree. An example is a private chat message or the event "avatar A gives avatar B item X".

multi-target Events with a multi-target context have a defined set of explicit recipients, for example a chat message to a group of participants who have a certain relationship in the virtual world or an event whose recipients are deduced by certain metrics. For example all avatars, which are targetable by one's cross-hair.

spatial An event with a spatial context is only relevant to a subset of recipients limited by spatial constraints. This class is a special case of the multi-target class. The distinction is necessary, due to the special optimized delivery a spatial context allows. An example for an event with a spatial context is the pickup of a flower in the virtual environment. Only clients in visual range need to be notified of such an event.

global An event which is broadcasted to all clients of the virtual world, without any restriction is of a global context. These events are distributed without any optimization.

2.1.2 Persistency

In contrast to normal multiuser virtual environments, MMVEs provide a persistent world, and therefore some events like e.g. the gain of money must be persistent in some way. Therefore we distinguish two classes for this dimension: transient and persistent events. There are two major solutions for the problem to persist events: Replication of the state, to ensure enough hosts are always online to restore the state or the storage in a centralized database [8].

2.1.3 Synchronization

Some event types have certain temporal or causal interdependencies and therefore require synchronization. For example, a position update may have no synchronization requirements, due to its high update rate. An event representing the pickup of an item from a chest on the other hand needs defined synchronization semantics, because there are causal interdependencies if another player also wants to pick up this item at the same time. There are many different approaches like *virtual time* [12, 9], all providing different synchronization semantics for different requirements. For this dimension, we have defined certain levels of synchronization, inspired by [13]:

weak This class of events does not need an explicit synchronization algorithm. It is not crucial to the virtual world, when and in which order an event arrives at the addressed nodes. For example, chat messages do not need an ensured order on each node, as their order is not essential to the operation of the chat service.

causal Causal synchronization defines a synchronized processing only for dependent events of an event type. In our domain two causal events of the same event type, generated by the same user have a defined relative order that has to be ensured by the algorithm. All not depending events of the same type may be processed in an arbitrary order on each node. That means explicit rules, modelling the dependencies between events of one type must be defined in this synchronization model. For example if two avatars are engaged in combat by targeting each other, their combat events have to be ordered, whilst all combat events from not targeted avatars are not causal dependent, as they have no impact on the two engaged avatars and therefore are not synchronized. The corresponding rule would be the mutual targeting avatars are causal dependent.

sequential Event types with sequential synchronization must follow two criteria: two actions of the same user must be processed in the correct order, whilst 2 actions of two different users must be processed on all nodes in a globally fixed order. For example, the event types "open chest" and "pickup content" have such synchronization requirements.

strict Strict synchronization defines, that all events of an event type have to be processed in the same order on all nodes. This defines a globally consistent and valid processing order for all events of that type. An example for such an event type is the "death of an avatar" event, whose order should be guaranteed.

2.1.4 Validity

Whilst synchronization describes the order of, or more generally the relationship between events of an event type, validity is strictly limited to one event, for example an effect on a player which is active for a certain time may be modelled by one event with the corresponding validity. This dimension decouples the time of the event delivery from the time an event's effect. E.g. events may be sent in advance describing an effect in the future, which means the event is valid only in a certain time interval in the future. We distinguish three characteristic classes of validity:

interval An event type may be valid for a certain time interval independent of its sending or its receive time. For example, a certain action triggers an effect for three minutes. With interval validity only one event is needed.

time-point An event type has an explicit point in time, at which events of this type are valid and to be processed.

For example, a scripted action of the world, like the crash of an airplane may be broadcasted ahead of time. Moreover, invalid events may be discarded to reduce messages in the system.

unlimited Unlimited validity means, this event type has a permanent impact on the virtual world. It must not be lost and therefore has to be delivered.

2.1.5 Delivery

Some events must reach their destinations, while others like position updates may have such a high frequency, that the loss of a single event does not cause any problems. Therefore the system may have to guarantee the delivery or prioritize it. Depending on different delivery characteristics, the systems may be optimized and reduce events.

guaranteed Event types with guaranteed delivery must reach their destinations. This may for example be ensured by an ack protocol.

prioritized Prioritized events create a possibility to ensure important events to be delivered, whilst optional or non time-critical events may be delivered later.

desired This class describes pure optional delivery with no guarantee.

2.1.6 Security

A secure event is not tempered and represents the initial event. Especially in distributed MMVE architectures, it is important to detect cheating clients at least. Prevention would be the optimum, but in most cases it is too expensive to guarantee cheat-free operation. Because of its impact on the performance of event dissemination, we see security of events as a semantically relevant dimension in this context. Therefore, we defined three classes: detection, prevention and no security.

All described dimensions provide aspects to classify events regarding their performance-relevant semantics and as all dimensions are deduced from existing approaches it is possible to optimize a system along these dimensions.

2.2 Examples of event semantics

Based on our dimensions with their characteristic classes, an event may be characterized along each dimension. We give some common event class examples based on an analysis of the events used in Quake 3 Arena. An exhaustive description of all potential classes is not possible at this point of our research as the analyzes of more games is needed to identify all relevant classes of event types. Following selected event types can be found in the Quake 3 Arena source code[4] and exemplary classified according to our classification:

Movement events have a spatial context, transient persistency, causal synchronization, interval validity, desired delivery and no security.

Jump events have a spatial context, transient persistency, causal synchronization, interval validity, guaranteed delivery and preventing security.

[4]Q3A 1.32b Source Code: `http://www.idsoftware.com/business/techdownloads/`

Item Pickup events have a spatial context, persistent persistency, sequential synchronization, unlimited validity, guaranteed delivery and preventing security.

Team Message events have a multi-target context, transient persistency, weak synchronization, unlimited validity, desired delivery and no security.

These examples show the variety of optimization potential each event type has and indicate the adequateness of our dimensions. The concrete classifications in a game may vary in detail, for example depending on the intended tolerance of the consistency in the virtual world. The consistency is mainly controlled by the synchronization semantics, but also influenced by physical constraints like network latency. This makes it even more important to provide different strategies optimized to the special requirements of each event type. Selected optimization strategies addressing different dimensions of our multidimensional classification schema are discussed in the next section.

3. OPTIMIZATION STRATEGIES

In recent years many promising approaches for optimization of event dissemination under exploitation of event semantics have been developed. We discuss selected approaches and categorize their optimization potential according to our multidimensional classification. It is important to point out that each of the mentioned approaches only addresses one or two of the introduced dimensions and therefore does not cope the full optimization potential event types may have.

3.1 Context optimization

Most work has been done in the field of context optimization, due to the fact that in a virtual world each avatar has a certain *Area of Interest* (AoI), which determines the area the avatar gets events for. SimMud [14] introduces regions and each region is controlled by one arbitrary node defined by the usage of a *Distributed Hash Table* (DHT). Many other approaches like [7, 19, 20] adapted this idea. VON [5] uses voronoi diagrams to partition the space of the virtual world. Only neighbors in the Voronoi space are directly connected and form an overlay network, which optimizes AoI messaging. Boulanger et al. [11] give an overview on the performance characteristics of different world partitioning algorithms like Delauny triangulation or Euclidean distance measures, which may be adapted to distributed context optimization. Müller and Gorlatc [15] propose another approach by introducing replicated regions, which each manage a subset of the entities in the whole region.

3.2 Persistency optimization

As an MMVE represents a persistent world, it is crucial to support event types with persistent effects. Zhang et al. [8] analyze different consistency approaches for persistence, for example snapshot mechanisms or distance based storage, which means only if a position change is greater than a certain threshold an update to the database is performed. Another approach is the usage of replication mechanisms like [18] to ensure a disconnected client's state can be restored. Which approach is more applicable mostly depends on the distribution architecture chosen and whether centralized elements are justifiable.

	Context	Persistency	Synchronization	Validity	Delivery	Security
SimMud [14]	√	√				
Colyseus [7]	√		√			
VON [5]	√					
Rokkatan [15]	√					
Donnybrook [6]	√					
Virtual time [12]			√			
Ferretti [9]			√			
Dead Reckoning [16]					√	
Kabus [17]						√
Mammoth [8]		√				
MiddleSIR [18]		√	√			

Table 1: Approaches and their optimization

3.3 Synchronization optimization

Colyseus [7] realizes a distributed architecture with one copy serialization for each entity in the virtual world under exploitation of contextual semantics. A more strict approach is *virtual time* [12] which introduces a virtual time for the virtual world and is able to keep a strict synchronization through reordering the events by the usage of negative events. Ferretti [9] enhances a virtual time approach by the usage of a gossiping protocol to speed up synchronization.

3.4 Validity and delivery optimization

The optimization of these two dimensions are classical event processing problems which may be addressed by acknowledgments on protocol level or with validity checks in each node. Specialized approaches in this field often focus on global message reduction like *dead reckoning* algorithms [16] or *update frequency control*, which prioritizes events to maintain a static bandwidth consumption.

3.5 Security optimization

The most optimized case is obviously to ignore security, but this is not feasible in the commercial sector. Despite common security measures like encrypting exchanged events, some research has already been done in this field like [17] which addresses different approaches to prevent or detect cheating.

3.6 Multi-purpose optimization

The described approaches for each dimension show a variety of optimization ideas which all have their advantages and drawbacks for certain specialized event types. As Table 1 shows, existing approaches are specialized only on a subset of our proposed dimensions, but MMVE architectures have optimization potential along all those dimensions. Each event type should be processed in the way, optimal for its type. This is only possible if the existing approaches are combinable in a modular fashion, depending on the event type. We strive for the exploitation of this manifold potential to take the optimizability of event dissemination architectures one step further. To achieve this goal, the depicted optimization techniques must be unified in a generalized and adaptable framework which is capable of using different event dissemination strategies depending on the class of the event type. The challenge regarding the combination and integration is the requirement of a general model to define such dissemination algorithms, which is part of our ongoing research. In the next section, we sketch a possible architecture, addressing the realization of such a system.

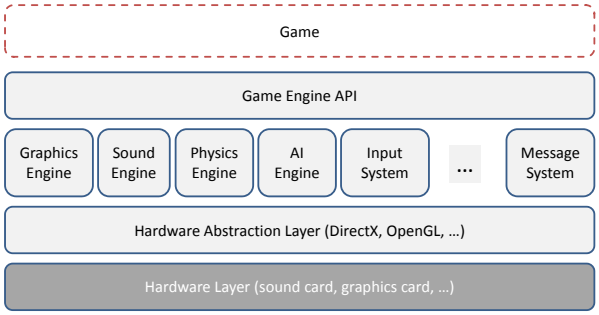

Figure 1: Coarse Game Engine Reference Architecture

4. ARCHITECTURE

The realization of an architecture supporting adaptable optimizations states many challenges. In this section, we identify the required modules to implement such a framework. But first brief discussion to the common architectural styles of an MMVE is given. This is required as a background in front of which a description of our architecture with its design decisions can be argued. In general, an MMVE consists of n interconnected nodes all being part of the same virtual world. This world is described by a global shared state, which is the sum of the states of all entities existing in the MMVE. Each node manages an as consistent as possible replica of this global state and runs a game engine which follows an architecture as depicted in Fig. 1. This game engine is responsible for the manipulation and the generation of a view on the world state. Fig. 1 shows a coarse reference architecture for such an engine. Based on the abstraction layer provided usually by the operating system, many subsystems are defined, each responsible for a different task in the engine. As we concentrate on the communication architecture and its optimization, when speaking of our framework, we talk about the *message subsystem* and its realization as a middleware.

Singhal and Zyda [16] identify the main technical challenge for networked virtual environments as the manage-

ment of their dynamic shared state. They also introduce two general approaches to cope with this challenge. On the one hand, a centralized approach may be chosen, holding all state information on one central server. On the other hand, all participating nodes in the MMVE may own parts of the dynamic shared state. As motivated in Section 1 we believe that distributed architectures will emerge as the future architectural style for MMVEs and therefore our architecture is based on a P2P overlay network.

The second architectural decision is the communication model between the nodes. There are two different models to describe the communication:

- **Query Update**
  For the computation of a new consistent game state, each client has to be informed of all changes of all other clients. Therefore, each client may request the corresponding update for its state replica. As a result, the game performs a fairly even mixture of queries and updates to ensure the consistency of each node's state. It is organized in bursts of $O(n)$ queries followed by $O(n)$ updates corresponding to the update rate of the game engine [2]. This processing style is more common to centralized architectures.

- **Publish/Subscribe**
  Some of the updates are only relevant for a subset of all participants of the game. Pub-sub systems offer a solution for topic-based optimization. Subscribers register themselves for a special topic. If a message appears for a topic, only subscribers of this topics are informed. In MMVEs, the list of recipients has to be computed because a recipient can not decide if he is a subscriber or not.

As there are many disadvantages for query/update communication in distributed environments, like the message overhead needed per frame to issue the queries, we use a distributed publish/subscribe messaging. Keeping the defined requirements in mind, it is obvious that even in publish/subscribe systems, to ensure global consistency for the state replicas of n nodes there is an effort of $O(n^2)$ required, if each node is authoritative for its own avatar. To optimize this effort to a reasonable level for supporting real-time MMVEs, one or more optimization techniques as introduced in Section 3 have to be applied. In our ***M**assive **M**ultiuser **E**vent **I**nfra**S**tructure* (m^{2}etis) project, we currently develop a framework to improve the event dissemination performance following our proposed multidimensional classification in Section 2.

4.1 m^{2}etis architecture

The m^{2}etis framework strives for the implementation of a networking middleware for MMVEs which integrates seamlessly as the messaging system into the reference architecture for game engines (cf. Fig. 1). It is designed around the central multidimensional semantic event classification, which makes the framework adaptable to many scenarios. Each MMVE has to provide a semantic classification of its event types according to our schema. Based on the definition of the type itself and its semantic properties the m^{2}etis system decides which algorithm is chosen to disseminate each event type across the participating nodes. One of the core benefits of this architecture is the separation of the game engine and the message dissemination architecture, which reduces the complexity in the game engine significantly.

Fig. 2 shows the modules of our architecture. The system consists of 3 parts: The m^{2}etis transformer responsible for mediation between game engine and the internal event model, the m^{2}etis optimizer which optimizes the event dissemination and creates the according optimized channels for its publish/subscribe system and the underlying P2P overlay network based on Chimera [21] a Tapestry advancement, which provides the routing capabilities for event distribution.

m^{2}etis transformer The transformer component consists of the formalized semantic model based on the multidimensional classification given in Section 2 and the m^{2}etis adapter. The model has to be instantiated for each application to reflect the semantic properties of the application's event types, meaning each event type used by the engine or the game has to be classified following the semantic model. The m^{2}etis adapter component provides the API to the other subsystems of the application engine. This API allows messaging and provides replicas or master copies of game states as required by the engine reference architecture. The adapter moreover transforms the states and messages of the API (states and events) to an internal event format which allows the m^{2}etis optimizer to calculate the optimized delivery properties.

m^{2}etis optimizer In this component the internal event representation is used to generate and operate optimized Publish/Subscribe multicast trees. Based on the semantic model and the internal event representation the dissemination optimizer deduces with the help of semantic application information, the recipients and the dissemination strategy for each event type. This happens based on the catalogue, provided by the optimization manager, which holds all optimization algorithms and their corresponding costs. Following a cost model, the dissemination optimizer is able to calculate the optimal algorithm of a certain event type. Based on this decision, the corresponding channel is created. The resulting optimized dissemination structure is managed by the channel manager and the subscription manager, which controls all moving subscriptions and the resulting changes in the dissemination structure as well as the dissemination of events itself.

Tapestry Based on the managed dissemination structures for each type, the channel manager is able to route each occurring event following to the associated algorithm and recipients. The routing itself is conducted by an underlying P2P Overlay-Network like Pastry [22] or Chimera [21], an advanced Tapestry implementation.

The challenges that arise from the depicted architecture are on the one hand the automation of the transformation steps to minimize the manual work. We aim for two inputs to deduce the optimized Publish/Subscribe multicast trees: A semantic model of the event types and a list of optimization algorithms with their costs. Whereby the algorithms and costs must not be adapted for every application, as a catalogue of supported algorithms is provided by the middleware. On the other hand the optimizer component with

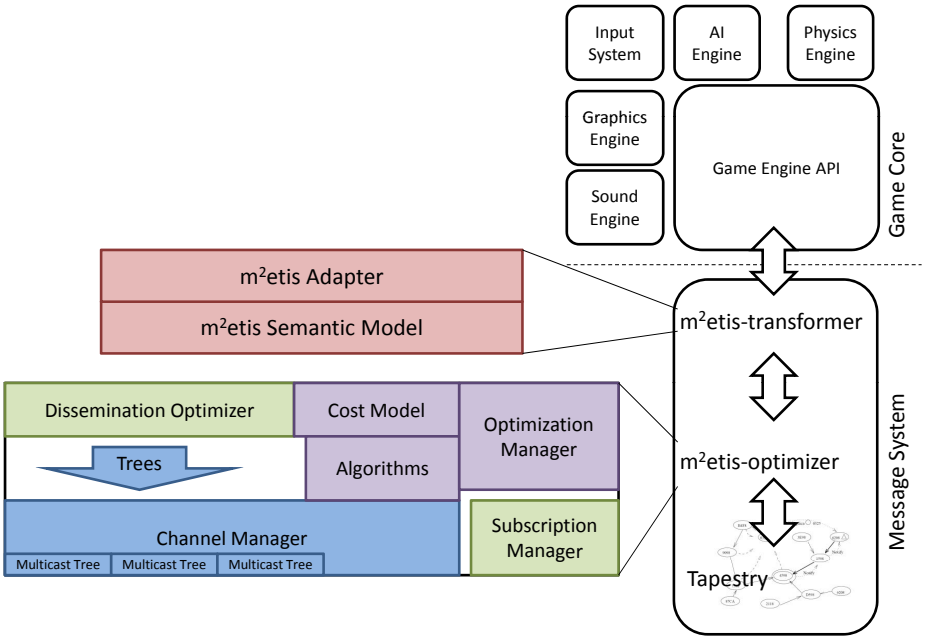

Figure 2: m2etis architecture

its cost model and formal reasoner is the other challenge to address.

5. CONCLUSION AND FUTURE WORK

Due to the huge number of players, MMVEs pose new challenges regarding the scalability of the underlying architecture. New holistic ways for minimizing the communication effort have to be found. We introduce the m^{2}etis project which aims for a generic architecture to optimize the amount of messages required in an MMVE without hampering quality requirements. In this paper, we proposed a multidimensional classification schema and outlined optimization strategies based on existing approaches.

We are currently implementing a prototype for the described architecture. The focus lies on the realization of a publish/subscribe system, capable of handling the required different event dissemination strategies and algorithms. The challenge is handling moving spatial subscriptions in the publish/subscribe component. To enable a generic extensible architecture which is able to integrate arbitrary optimization algorithms, a generalization of the optimization strategies is required. Moreover a formal semantic model based on the proposed classification has to be developed. Complementing it with a cost model, enables an optimizer component that allows automated customized optimization for application-specific requirements.

Existing approaches only cope with a subset of our identified dimensions, therefore our idea to compose different optimization strategies based on our multidimensional semantic classification poses a holistic approach. We understand the work presented in this paper as a first step on the road to generic event dissemination systems which are optimizable based on the knowledge of application-driven event semantics.

6. REFERENCES

[1] Reinhard Müller and Frank Mackenroth. German Entertainment and Media Outlook: 2006-2010. PriceWaterhouseCoopers (Report), October 2006.

[2] Walker White, Christoph Koch, Nitin Gupta, Johannes Gehrke, and Alan Demers. Database Research Opportunities in Computer Games. *ACM SIGMOD Record*, 36(3):7–13, 2007.

[3] Max Skibinsky. *Massive Multiplayer Game Development 2*, chapter 5 - The Quest for the Holy Scale-Part 1: Large-Scale Computing, pages 339–355. Game Development Series. Charles River Media, 2 edition, February 2005.

[4] Halldor Fannar, Victoria Coleman, Randy Breen, and Brandon Van Slyke. The server technology of eve online: How to cope with 300,000 players on one server. Talk on GDC Austin, 2008.

[5] 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 2006.

[6] Ashwin Bharambe, John R. Douceur, Jacob R. Lorch, Thomas Moscibroda, Jeffrey Pang, Srinivasan Seshan, and Xinyu Zhuang. Donnybrook: Enabling Large-Scale, High-Speed, Peer-to-Peer Games. *SIGCOMM Comput. Commun. Rev. (CCR)*, 38(4):389–400, 2008.

[7] Ashwin Bharambe, Jeffrey Pang, and Srinivasan Seshan. Colyseus: A Distributed Architecture for Online Multiplayer Games. In *3rd Symposium on Networked Systems Design & Implementation (NSDI)*, pages 155–168, Berkeley, CA, USA, 2006. USENIX Association.

[8] Kaiwen Zhang, Bettina Kemme, and Alexandre Denault. Persistence in Massively Multiplayer Online Games. In *7th ACM SIGCOMM Workshop on*

Network and System Support for Games (NetGames), pages 53–58, New York, NY, USA, 2008. ACM.

[9] Stefano Ferretti. A Synchronization Protocol For Supporting Peer-to-Peer Multiplayer Online Games in Overlay Networks. In *2nd International Conference on Distributed Event-Based Systems (DEBS)*, pages 83–94, New York, NY, USA, 2008. ACM.

[10] Thomas Fischer and Richard Lenz. Event semantics in event dissemination architectures for massive multiuser virtual environments. In *DEBS '10: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, pages 93–94, New York, NY, USA, 2010. ACM.

[11] Jean S. Boulanger, Jörg Kienzle, and Clark Verbrugge. Comparing Interest Management Algorithms for Massively Multiplayer Games. In *5th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, New York, NY, USA, 2006. ACM. Article No. 6.

[12] David R. Jefferson. Virtual Time. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 7(3):404–425, July 1985.

[13] Rivka Ladin, Barbara Liskov, Liuba Shrira, and Sanjay Ghemawat. Providing High Availability Using Lazy Replication. *ACM Transactions Computer Systems*, 10(4):360–391, 1992.

[14] B. Knutsson, Honghui Lu, Wei Xu, and B. Hopkins. Peer-to-Peer Support for Massively Multiplayer Games. In *IEEE INFOCOM*, volume 1, pages 96–107. IEEE, 2004.

[15] Jens Müller and H. Sergei Gorlatc. Rokkatan: Scaling an RTS Game Design to the Massively Multiplayer Realm. *ACM Computers in Entertainment (CIE)*, 4(3):1–14, 2006. Article No. 11.

[16] Sandeep Singhal and Michael Zyda. *Networked Virtual Environments: Design and Implementation.* Addison-Wesley Professional, New York, NY, USA, 1999.

[17] Patric Kabus, Wesley W. Terpstra, Mariano Cilia, and Alejandro P. Buchmann. Addressing Cheating in Distributed MMOGs. In *4th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, New York, NY, USA, 2005. ACM. Article No. 6.

[18] Yi Lin, Bettina Kemme, Marta P. Martinez, and Ricardo J. Peris. Applying Database Replication to Multi-player Online Games. In *5th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, New York, NY, USA, 2006. ACM. Article No. 15.

[19] Shinya Yamamoto, Yoshihiro Murata, Keiichi Yasumoto, and Minoru Ito. A Distributed Event Delivery Method with Load Balancing for MMORPG. In *4th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, New York, NY, USA, 2005. ACM. Article No. 8.

[20] Thorsten Hampel, Thomas Bopp, and Robert Hinn. A Peer-to-Peer Architecture for Massive Multiplayer OnlineGames. In *5th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, New York, NY, USA, 2006. ACM. Article No. 48.

[21] Matthew S. Allen and Rame Alebouyeh. Chimera: A Library for Structured Peer-to-peer Application Development. Technical report, University of California, Santa Barbara, 2006.

[22] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350. Springer-Verlag, 2001.