Budget Based Dynamic State Update Aggregation

Lutz Behnke[1], Qi Wang[2], Christos Grecos[2], and Kai von Luck[1]

[1]Dept. of Computer Science, University of Applied Sciences Hamburg
[2]School of Computing, Univ. of the West of Scotland, Paysley, Great Britain

ABSTRACT

Distributed Virtual Environments (DVEs) can handle large numbers of participants by optimizing the routing of state update messages sent to the computer of the individual participant. Often Area of Interest (AoI) management is used to limit the information an avatar can observe and thus the necessary state update messages that are sent to the client which the avatar represents.

If the avatars are roughly evenly distributed through out the virtual environment, AoI enables the support for very large numbers of participants. But once a sufficient number of avatars gather in a small area of the virtual space, communication overhead grows on a n^2 scale as they observe each other.

Instead on optimizing the overall number of messages, we limit the number of messages sent to a single client. We propose a novel approach to dynamically aggregate state update messages. It can support a very high numbers of avatars in restricted spaces. We decouple the management of the global state of the DVE from amount of detail sent to the client. Our approach exploits information about the specific DVE and its current state to maximize the ratio of bandwidth reduction to state update fidelity. It is intended to provide rich user experience for clients that connect via limited network links, like mobile users or consumer grade DSL.

Categories and Subject Descriptors

C.2 [**COMPUTER-COMMUNICATION NETWORKS**]: Distributed Systems

General Terms

Algorithms

MMVE'15 March 18-20 2015, Portland, OR, USA

ACM 978-1-4503-3354-2/15/03 ...$15.00
http://dx.doi.org/10.1145/2723695.2723696

Keywords

DVE, MMOG, Mobile

1. INTRODUCTION

Multiple techniques for managing Areas of Interest (AoI) and filtering state change messages based on these areas have been proposed in recent years. A good overview is provided by [3]. Approaches for client/(multi-)server, as well as peer-to-peer architectures have been proposed. Independent of the underlying architecture the usually aim to reduce the state update messages sent to a given client by limiting them to those that can actually be observed by an avatar and thus are required by the client.

[4] considers the need to verify MMO Architectures for varying avatar concentration and movement patterns. From his work, we take the concept of the Avatar Density (AD), the number of avatars in a given spatial area. The global AD represents the total number of client connections and thus avatars supported by a DVE. In contrast we focus on the maximum supported local AD, the number of the avatars in high density situation. Work like [5] has shown that the tight clustering of human players is the norm rather than the exception.

For a high local AD the nimbus' of the avatars overlap and they can each see each other, resulting a n^2 communication overhead. Handling this by dynamically reducing the observation range may render avatars effectively blind. It also results in a restriction of the avatars abilities due to technical reasons and is likely to be perceived as a service degradation by the user.

We present Dynamic Budget Based State Aggregation (*DyBubSA*), a mechanism for fast discovery of applicable aggregators from a repository to handle local overload scenarios. While most aggregators are based on common algorithms as reducing the update ratio or the detail of information sent, *DyBubSA* allows the integration of DVE specific aggregators, that exploit elements of the content within the DVE (see section 2.2 for some examples).

We also introduce a novel way to select applicable aggregators and apply them to specific avatars and their context.

Our work is complementary to aggregate network packets, like [2], that allow the combination of multiple messages into single network packets.

2. APPROACH

The prototype of *DyBubSA* is based on QuP ([1]), a system for persisting a unified DVE state, distributing state

updates to networked game servers, using eventual consistency to provide resilience to component failure. Figure 1 shows the software elements of a single instance of a set of game server nodes.

The aggregators behave as avatars, subscribing to state updates that are element of the aggregate. The aggregators will only actually subscribe to update messages and process these if at least one avatar or other object (e.g. another aggregator) is subscribed to the event stream that the aggregator produces. QuP makes this simple, as the list of subscribed entities is a published attribute available to other subscribers.

Upon activating an aggregator, the avatar will be informed of the covered objects, unsubscribe from the objects and subscribe to the avatar. By subscribing to it, the aggregator will be activated, should it have been dormant.

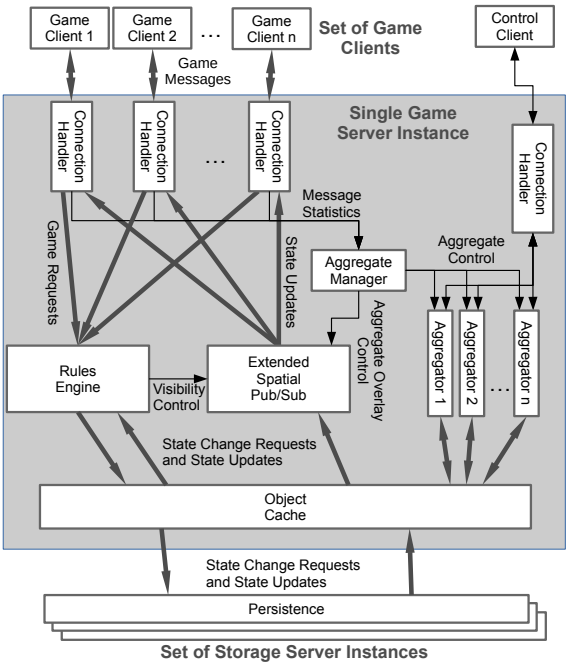

Figure 1: software components of system

2.1 Budget

For each client connection a message budget is set. Appropriate values of connection can be obtained from various sources, e.g. a) an arbitrary value set by the DVE operator, e.g. to limit bandwidth costs; b) information on end-to-end network bandwidth, as congestion messages or an IP address block assigned to mobile service provider.

DyBubSA tracks the message rate sent to each client. For our prototype the length of each message is fixed and thus the number of messages provides a sufficient approximation of the required bandwidth.

2.2 Types of aggregators

DyBubSA supports two basic classes of aggregators. The first are global aggregators, which are applicable to any object and can be dynamically instantiated and applied to any object generating update messages. They usually support sending fewer updates per time or ignoring certain types of updates. The most basic of these aggregators simply forwards only every nth update message.

The second class of aggregators are application specific and support the collation of multiple objects into a compound object that generates resultant update messages on a much more coarse level. Good examples for this are tight military formations of many soldiers, acting as a single entity. Or a train composed of a locomotive and a number of carriages.

2.3 Aggregator Selection

The compound aggregators are positioned according to their spatial dimensions as any other object managed by QuP. The resultant axis aligned bounding box (AABB) is updated by the aggregator according to the distribution of the objects it covers. By traversing the octree downward from the node representing the vision range of an object, applicable aggregators can be found. By choosing the applicable aggregator with the greatest distance to the avatar, the impact on the user experience can be minimized as well, as details further away are displayed smaller and are not as likely to be in the focus of the user as much.

Each aggregator also publishes a message reduction ratio as as well. Additional aggregators are selected until the projected aggregates message rate is below the intended rate.

Global aggregators are registered with the top node of the octree and will perform their filtering for each avatar that subscribes to them instead of a specific original object.

3. SUMMARY

We are currently evaluating a prototype for this approach to support very large sets of avatars in potentially close proximity. This is work and progress.

4. REFERENCES

[1] L. Behnke, C. Grecos, and K. V. Luck. QuP : Graceful Degradation in State Propagation for DVEs. In *Proc. of MMVE 2014*, Singapore, 2014. ACM.

[2] W. Jinzhong. On packet aggregation mechanism for improving Massive Multiplayer Online Game performance in P2P network. *2011 3rd International Conference on Computer Research and Development*, pages 337–339, Mar. 2011.

[3] H. Liu, M. Bowman, and F. Chang. Survey of state melding in virtual worlds. *ACM Computing Surveys*, 44(4):1–25, Aug. 2012.

[4] P. Morillo, J. M. Orduna, and J. Duato. A scalable synchronization technique for distributed virtual environments based on networked-server architectures. In *Proceedings of the International Conference on Parallel Processing Workshops*, pages 74–81. RAND Europe/Ofcom, 2006.

[5] S. Shen and A. Iosup. Modeling Avatar Mobility of Networked Virtual Environments. In *Proceedings of International Workshop on Massively Multiuser Virtual Environments (MMVE'14)*, pages 1–6, Singagapore, 2014. ACM Press.