Event processing for large-scale distributed games

Gerald G. Koch Muhammad Adnan Tariq Boris Koldehofe Kurt Rothermel

Universität Stuttgart
Institut für Parallele und Verteilte Systeme
Universitätsstr. 38
D-70569 Stuttgart, Germany
{firstname.lastname}@ipvs.uni-stuttgart.de

ABSTRACT

Novel peer-to-peer-based multiplayer online games are instantiated in an ad-hoc manner without the support of dedicated infrastructure and maintain their state in a distributed manner. Although their employed communication paradigms provide efficient access to sections of distributed state, such communication fails if the participants need to access large subsets of the application state in order to detect high-level situations. We propose a demonstration that shows how multiplayer online games can benefit from using publish/subscribe communication and complex event processing alongside their traditional communication paradigm.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organisation**]: Distributed Systems

General Terms

Multi-player online game, content-based publish/subscribe, distributed complex event processing

1. INTRODUCTION AND MOTIVATION

A novel kind of large-scale applications that users instantiate in an ad-hoc manner requires new ways of maintaining and accessing application state. For instance, peer-to-peer-based multiplayer online games avoid a dedicated infrastructure of game servers and keep the game state in a distributed manner, which introduces new challenges. For example, correct game behaviour requires that interacting participants have a consistent view on the game's state. Therefore, efficient communication paradigms are essential—e.g., group communication with clustering schemes [3]—for the update and access to distributed application state.

However, in realistic scenarios, such communication schemes are not sufficient. First, while the accessible game state contains only low-level representation like the position of an object, players are rather interested in more abstract *situations*. For example, the gathering of a certain amount of players or the development of a specific setting are not represented as a value in the games' state. They require the detection of temporal, spatial or other patterns in the behaviour of objects. Furthermore, letting all players access the global state and detect situations on their own creates an unnecessary load for communication and computation. Traditional communication paradigms cannot handle this problem as they typically have no event processing capabilities.

Second, players are usually interested in situations that occur within a defined area called the *area of interest* (AOI). However, communication paradigms like group communication have typically no access to the semantics of the game and therefore run clustering schemes that ignore the player's AOI and thus lose efficiency. Additionally, group communication schemes need to keep groups small and distinct which leads to multicast group explosion if players are interested in a large number of diverse sections from the distributed application state. As a result, realistic scenarios in multiplayer online games that are currently supported by traditional communication paradigms require additional support for the the observation of individual high-level situations.

2. APPROACH

The demonstration shows that peer-to-peer-based multiplayer online games benefit from additional support by content-based publish/subscribe and distributed complex event processing. With this approach, instances of the game application publish selected parts of the game state to the content-based publish/subscribe system *EONSON* [2], in addition to the group communication system. *EONSON* provides efficient state dissemination as it incorporates the game semantics of "virtual space" directly in its tree structure and dissemination process. For instance, if the game instance publishes virtual location information, this information is employed by the content-based publish/subscribe system in order to coordinate the subscribers on a peer-to-peer basis. The result is a publish/subscribe overlay topology that reflects the containment relationship between the AOI of game instances which publish location information or subscribe to it. The topology ensures that only peers that are interested in the same areas participate in forwarding and filtering messages, providing an efficient event distribution throughout the whole distributed game application.

In addition to content-based event communication, the distributed event correlation detection system *Cordies* [1] is employed. Its instances are located independently from each other within the distributed application. In particular, their

location does not depend on the message flow of the publish/subscribe system. The *Cordies* instances subscribe to event messages and detect several application-defined event patterns on them. Detected patterns lead to the publication of corresponding complex event messages. The *EONSON* publish/subscribe system forwards complex event messages to other *Cordies* instances or to game instances subscribing to those situations. The application defines event patterns separately from subscriptions, which allows a fine-tuning in the levels of abstraction. For instance, one event pattern detects meetings of any pair of players in a defined area of the virtual game space. Players, however, subscribe to a more restricted situation—the meeting of two players where they represent one party. The situation detection is executed once, but all players receive different user-defined information thanks to message filtering in *EONSON*.

Supplementing the communication paradigm employed by a peer-to-peer based multiplayer online game with *EONSON* and *Cordies* yields the following benefits. First, content-based communication uses message contents for information forwarding and is therefore capable of considering application semantics when restructuring which leads to more efficient communication. Still, the instances of the game that publish game state or subscribe to it, stay decoupled. Second, employing distributed complex event processing which blends seamlessly into the publish/subscribe system provides useful high-level situation detection which is accessed by game instances using the common publish/subscribe interface. This relieves the game instances from detecting situations themselves—possibly repeatedly at several instances. Furthermore, the *Cordies* can regulate the placement of its instances autonomously and provides in-network complex event processing, thus reducing bandwith consumption.

3. DEMONSTRATION

The demonstration uses the multiplayer first person shooter game *PlanetPI4* [3] to show the benefit of in-network event processing. The game has a peer-to-peer architecture and manages its state information in a distributed manner. Each player controls a plane in the virtual game space and can shoot other planes if they are within the shooting range.

Planes within shooting range may not always be in the focus of the player (such as behind the player's plane). Therefore, a player should receive a notification whenever another plane is in its AOI. Detecting this situation locally would require a lot of state information at each player, which does not scale in terms of communication, storage and computational overhead. However, with in-network detection of the situation (Figure 1), *positionUpdate* messages of planes are forwarded to *Cordies* nodes rather than disseminated to the network boundary. These messages are interpreted by *Cordies* nodes in three different ways. First, two position messages of the same plane describe a movement. The node publishes an *onMovement* message accordingly, which is not yet of interest to the game application. Second, a *positionUpdate* message of one plane and a *onMovement* message of another can describe the meeting or the departure of two planes. Players can subscribe to *onMeeting* and *onDepart* messages and react accordingly.

Maintaining a single *Cordies* instance for testing meetings and departures may still not scale with the number of players. Here, the application semantics of the area of interest (AOI) is handy. Communication and computation overhead

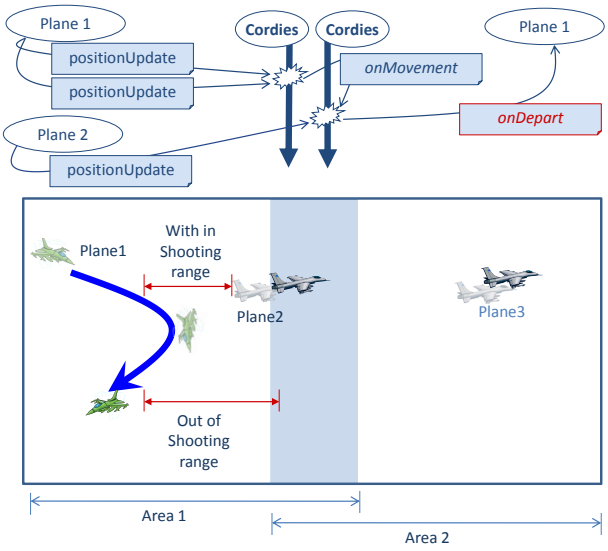

Figure 1: Demonstration Scenario

can be reduced if the game divides its virtual space into static AOIs, so that state information is provided completely and fast to all the planes within an AOI. For scalability reasons, a separate *Cordies* instance detects event patterns for each AOI. Each instance subscribes to *positionUpdate* messages from the assigned AOI.

The table below compares the demonstrated event service and a clustering multicast service with respect to the estimated number of delivered messages and performed correlation tests per time unit. It assumes that the event service has c static AOIs and that multicast has c clusters, with an equal number of nodes in each AOI or cluster, and that each of the total n nodes sends a *positionUpdate* per time unit. The min/max values stand for the minimal/maximal number of possible occurring situations. The demonstrated solution has less message overhead in most cases and less correlation test overhead if approximately $c < \frac{n}{3}$.

	multicast	**demonstrator**
Nr. of messages	$n\left(\frac{n}{c}-1\right)$	min: n max: $n\left(\frac{n}{c}+1\right)$
Nr. of tests	min: $\frac{n^3}{c^2}-\frac{2n^2}{c}+n$ max: $\frac{n^3}{c^2}-n$	min: $\frac{n^2}{c}$ max: $\frac{3n^2}{c}$

4. REFERENCES

[1] G. G. Koch, B. Koldehofe, and K. Rothermel. Cordies: Expressive event correlation in distributed systems. In *Proc. 4th Intl. Conf. on Distributed Event-Based Systems (DEBS'10)*, 2010.

[2] M. A. Tariq, G. G. Koch, B. Koldehofe, I. Khan, and K. Rothermel. Dynamic publish/subscribe to meet subscriber-defined delay and bandwidth constraints. In *Proc. 16th Intl. Conf. on Parallel Computing (Euro-Par)*, 2010.

[3] T. Triebel, B. Guthier, T. Plotkowiak, and W. Effelsberg. Peer-to-peer voice communication for massively multiplayer online games. In *Proc. 6th IEEE Consumer Communications and Networking Conf. (CCNC)*, 2009.