Performance Evaluation of Peer-to-Peer Gaming Overlays

Max Lehn* Tonio Triebel† Christof Leng* Alejandro Buchmann* Wolfgang Effelsberg†

∗ Databases and Distributed Systems
Technische Universität Darmstadt
Germany

{max_lehn, cleng, buchmann}@dvs.tu-darmstadt.de

† Praktische Informatik IV
Universität Mannheim
Germany

{triebel, effelsberg}@pi4.informatik.uni-mannheim.de

***Abstract*—In this demo we present a performance evaluation testbed for peer-to-peer gaming overlays. It consists of a 3D first person shooter game that is designed to run in a simulated network environment as well as on a real network. Simulation with autonomous players (bots) guarantees scalability, a controlled workload, and reproducible results; a prototype deployment on a real network can then validate the simulation results. The information dissemination overlay pSense is implemented as a first subject for evaluation.**

I. Introduction

In the last years gaming has become an attractive field for peer-to-peer research. In particular the strict timing requirements of first person shooter games (FPS) demand the development of suitable network overlays. The main challenge consists of the combination of high dynamic movement of players and low latencies that are required for update messages. Specific peer-to-peer overlays, such as pSense [1], VON [2], or Donnybrook [3], have been developed to address these needs. For the evaluation of their systems most research groups use simple custom game simulations. Only a few overlays were analyzed with real games.

The simulation of a simplified game is hard to validate. The simulator might abstract away important details that affect the outcome of the simulation significantly. On the other hand, a standalone application cannot compete against a simulator in terms of parameter flexibility, reproducibility, and precise measurement.

We propose a benchmark platform for the evaluation of peer-to-peer gaming overlays that brings the two approaches together. The platform is based on the 3D multiplayer spaceship shooter Planet π4 [4]. A key feature of our system is the ability to run the game both as a standalone application using the real network and in a discrete-event simulation on a virtual network. Hence, it is possible to evaluate an overlay in a discrete-event simulation with a workload generated by autonomous players (bots) and to validate the results with the prototype deployment using a real network. In addition, our system provides the possibility to replace the peer-to-peer overlay implementation in order to compare different overlays. For the demonstration an implementation of pSense is used to show the two main functionalities of our system.

Fig. 1. A screenshot of Planet π4

II. pSense

pSense [1] is an information dissemination overlay addressing the specific need for exchanging updates of game states based on virtual world proximity. Each player has a *vision range* which determines the interest of activities of other players (e.g. movements and shots). Thus, in the pSense topology a node knows all neighbors in its vision range. To prevent network partitions in sparse density areas, each node keeps a list of eight *sensor nodes* outside its vision range. These nodes also introduce new nodes approaching the vision range.

In its original publication, like many other peer-to-peer systems, pSense was only evaluated with simulations lacking most networking detail. This work is the first to show that pSense works in a real game application.

III. Implementation

In standalone application mode, like in conventional game implementations, the game's main loop repeatedly renders a frame, processes user input, runs the game mechanics, and performs network I/O. The frame rate is only limited by the hardware capabilities. In simulation, the hardware is abstracted, and the behavior is mapped to discrete-event mode. The simulated instances usually do not render frames since

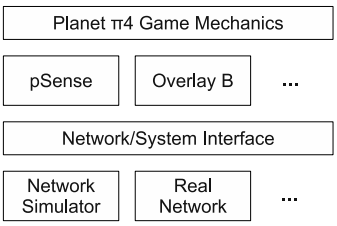

Fig. 2. High level system architecture

there is no user who needs to see them. For watching the simulation, frame rendering can still be enabled. A frame is then rendered each 100 simulated milliseconds. Thus, the game has a virtual frame rate of 10 frames per second.

Planet π4 and the pSense implementation are written in C++, while the simulator and network subsystem which we are currently using come from the BubbleStorm [5] project[1] and are written in Standard ML. To switch between simulation and real application all that is necessary is to replace the small component controlling the main loop in Planet π4's core. All other components remain unchanged, provided that they fulfill certain requirements. Most importantly, I/O and wait operations are restricted to pure asynchronous mode, since blocking calls cannot be executed in a discrete-event simulation.

IV. Workload: Bots

The measurement of overlay properties in the game requires appropriate workloads. The obvious way to generate workload is using human players. But setting up a testbed environment for human players takes a lot of effort, especially in larger scales (100 or more players). Furthermore, the generated workload depends on the behavior of the players and thus is not reproducible. The alternative of using traces from human player matches is not convincing, since the highly interactive gameplay is influenced by system properties (e.g., latency and consistency). The traces measured on one system cannot represent a realistic workload on a different system.

We chose to use bots to generate reproducible and easily scalable workloads. For the currently very simple gameplay, the bot implementations have simple aim-and-shoot mechanisms. More complex gameplay modes of course need more complex bot implementations. Those will then have to be tuned to an activity level similar to human players.

V. Demonstration

The demonstration consists of two parts: a physical local area network and a simulated Internet game.

A. Real-World LAN Game

A small set of PC runs the application in standalone mode, joining a common game session. The PCs are available for human players to play against each other and against the bots which can be added to the session. Status information like network traffic and overlay neighbor lists is made available through an in-game head-up display.

[1]http://www.bubblestorm.net

B. Simulated Internet Game

The discrete-event simulation runs several game instances (each controlled by a bot) within one operating system process. The number of instances in the simulation is limited by the available amount of RAM; 4GB currently suffice for up to 128 instances. The game can be watched from the perspective of one of the instances.

VI. Conclusion and Future Work

Our system provides a reproducible setup in a discrete-event simulation with a workload generated by bots and prototype deployment for real networks with a workload generated by human players and/or bots, both using the same peer-to-peer system implementation. Planet π4 provides the base for a realistic benchmark for peer-to-peer systems focusing on realtime capabilities. And as a game Planet π4 is still simple enough to concentrate on the core aspects.

For the further evaluation of peer-to-peer overlays we plan to extend Planet π4's gameplay, replacing the simple deathmatch mode with a team mode in which the teams have to defend certain strategic points of interest. The richer gameplay is supposed to increase the realism of the game workload, the fun factor for human players, and the scalability (number of players) of the gameplay.

We like to extend our measurement infrastructure to the standalone mode. While system-wide aggregation of statistics is trivial in the simulation environment, the distributed nature of the standalone mode makes the aggregation of highly timing-sensitive statistics challenging.

Finally, we want to add support for the widely used ns-3[2] network simulator as a third option in addition to our custom simulator and the real network stack.

This demo proposal and the related implementations where developed in the QuaP2P research group[3] funded by the Deutsche Forschungsgemeinschaft (DFG).

References

[1] A. Schmieg, M. Stieler, S. Jeckel, P. Kabus, B. Kemme, and A. Buchmann, "pSense-Maintaining a Dynamic Localized Peer-to-Peer Structure for Position Based Multicast in Games," in *P2P'08. Eighth International Conference on Peer-to-Peer Computing*, 2008, pp. 247–256.

[2] S.-Y. Hu and G.-M. Liao, "Scalable Peer-to-Peer Networked Virtual Environment," in *NetGames '04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, 2004, pp. 129–133.

[3] A. Bharambe, J. R. Douceur, J. R. Lorch, T. Moscibroda, J. Pang, S. Seshan, and X. Zhuang, "Donnybrook: Enabling Large-Scale, High-Speed, Peer-to-Peer Games," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 389–400, 2008.

[4] T. Triebel, B. Guthier, R. Süselbeck, G. Schiele, and W. Effelsberg, "Peer-to-Peer Infrastructures for Games," in *NOSSDAV '08: Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2008, pp. 123–124.

[5] W. Terpstra, J. Kangasharju, C. Leng, and A. Buchmann, "BubbleStorm: Resilient, Probabilistic, and Exhaustive Peer-to-Peer Search," in *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, 2007, pp. 49–60.

[2]http://www.nsnam.org/

[3]http://www.quap2p.tu-darmstadt.de/