Lamoth: A Message Dissemination Middleware for MMOGs in the Cloud

Julien Gascon-Samson
School of Computer Science
McGill University
Montreal, Canada
julien.gascon-samson@cs.mcgill.ca

Bettina Kemme
School of Computer Science
McGill University
Montreal, Canada
kemme@cs.mcgill.ca

Jörg Kienzle
School of Computer Science
McGill University
Montreal, Canada
joerg.kienzle@mcgill.ca

***Abstract*—Provisionning network resources for Massively Multiplayer Online Games (MMOGs) poses interesting challenges due to the fact that the load can greatly vary depending on the time or other in-game factors. In this paper, we propose Lamoth, a cloud middleware for MMOGs that provides an interface for in-game message dissemination. Lamoth handles the exchange of game messages between nodes by making use of an arbitrary number of off-the-shelve pub/sub servers deployed in the cloud depending on the game scenario. In order to evaluate our platform, we implement Lamoth on top of Mammoth, McGill's research-oriented MMOG, and conduct extensive experiments by triggering situations which would cause networks bottlenecks. Our evaluations show that Lamoth can allow a MMOG to scale to high numbers of players and can properly handle extremely-demanding in-game situations if enough resources are provided.**

I. Introduction

Massively Multiplayer Online Games (MMOGs) are a multi-billion dollar industry and feature thousands of players participating in shared huge virtual environments. To support such games, a powerful network infrastructure is needed, especially regarding the *message dissemination service*, which is responsible for delivering messages between the clients and the server(s). As players are more likely to be playing at certain time periods, the number of messages to be transmitted is subject to high variation over time. Also, a game provider might organize events at specific game locations which may attract many players in a confined area. In this paper, we argue that the cloud can be leveraged in the context of MMOGs to manage large numbers of connected clients under various scenarios which are likely to put a stress over the network infrastructure, such as a high volume of players and many players *flocking* towards a common location.

We propose Lamoth, a middleware platform used by all game nodes to route all game messages through a set of off-the-shelve pub/sub servers deployed in the cloud. In our implementation, we decided to use Redis, a popular ready-to-use open-source cloud middleware which provides, amongst other things, efficient pub/sub capabilities.

II. Related Work

Many approaches have been proposed to deal with server overloading issues such as sharding, approaches based on geographical mapping [1] and approaches based on logical mapping [2]. Some other approaches try to scale using multicast trees, at the expense of increased latencies [3]. Some popular approaches nowadays such as OnLive [4] run the complete game on servers. Players receive a video stream of the game and send back player input. The downside is that it requires huge amounts of bandwidth (server-side outgoing bandwidth and client-side incoming bandwidth). In [5], the authors describe how they deployed and made use of a pub/sub infrastructure in the cloud to support FPS games with large amounts of players. The architecture is however different.

III. Background

We conducted extensive experiments with Mammoth, McGill's research-oriented Multiplayer and Massively Multiplayer Online gaming platform [6]. We observe that in many cases, game servers get saturated due to limited network bandwidth and that other system resources such as CPU and memory were not fully used. Lamoth, aims at providing an efficient "message dissemination service" that leverages resources that the cloud can provide by making use of an already-available pub/sub middleware such as Redis.

Our experiments also reveal that a game transmits one of three kinds of messages: *unicast* (one recipient), *publication* (all subscribers of a given channel) and *broadcast* (all nodes). In addition to that, there must be *subscribe* and *unsubscribe* operations to register/unregister nodes to channels. Following our observations, we claim that the publish/subscribe paradigm can be used to effectively model message transmission in MMOG games because most message transmissions will be publication messages. The typical use of pub/sub operations is to perform state dissemination to interested entities in combination with interest management techniques. However, despite using such techniques, the required bandwidth will grow higher than the available bandwidth under some conditions. We denote three of such situations: too many connected players, too many players in the same area (*flocking*) and players increasing the rate at which they perform actions.

IV. System Architecture

Lamoth interfaces with MMOGs and provides a message dissemination service. It has two components: a thin software layer (library) that take as input all messages to be transmitted (unicast, publication, broadcast) as well as the subscribe/unsubscribe operations and a set of (at least one) publish/subscribe (p/s) servers that are to be deployed in the cloud (public or private cloud). Pub/sub cloud servers are only running Redis or some other middleware and no

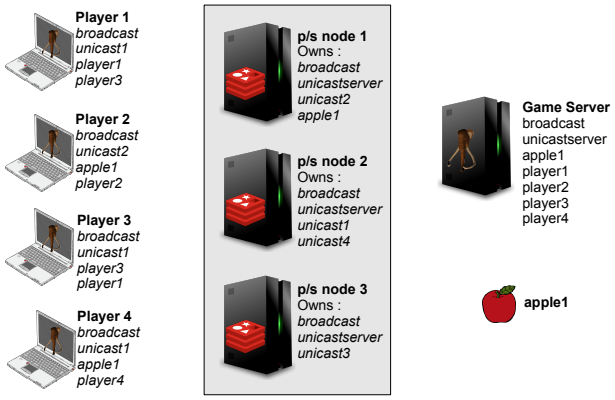

Fig. 1. Game scenario example showing 4 players, 3 p/s servers, one game server as well as a non-player object. All game nodes use the Lamoth software layer to handle the transmission of all messages. Lamoth which makes use of the the 3 p/s nodes to handle the delivery process.

other custom/game specific code. Furthermore, p/s servers are independent, non-clustered and do not communicate with each other. Lamoth employs a flat pub/sub architecture instead of a layered architecture in order to optimize latency.

The goal of the software layer is to provide a bridge between every node in the system and the set of p/s servers by handling all message transmissions. The software layer transforms all message transmission calls into pub/sub operations and dispatches them to the appropriate p/s server(s) who dispatches them to the intended recipients. The Lamoth software layer also receives incoming messages, decodes them and queues them to be processed by the game. Pub/sub nodes only have to support the three main pub/sub operations: *publish*, *subscribe* and *unsubscribe*.

Because Lamoth is able to handle the exchange of all messages between all nodes using only basic pub/sub middleware, it becomes possible to scale by deploying additional pub/sub (p/s) servers. Lamoth employs a circular hashing technique to map *channels* to p/s servers in order to split the load evenly on all servers. However, servers must be statically defined prior to launching the game since we do not yet support adding/removing p/s servers dynamically. There is ongoing research towards integrating this capability. Figure 1 gives an example of a game making use of the Lamoth platform.

V. System Implementation and Experiments

In order to test the scalability of our proposed platform, we developped a new network engine for the Mammoth framework that fully implements the Lamoth platform. We also made use of the Redis cloud middleware which supports pub/sub primitives. We ran our experiments on a pool of 100-150 typical desktop machines from the McGill School of Computer Science labs. Since cloud providers typically use a large pool of commodity machines, we believe that our setup can yield results that are similar to experiments run on real cloud provider infrastructure. Clients are controlled by AI which imitates real player movements (NPCs). Our deployment setup consists of a set of nodes simulating clients, a set of nodes for hosting Redis instances and one node for the game server. All communications go through the Redis nodes.

We ran experiments that attempted to connect as many players as possible as well as *flocking* experiments, combined with extensive parameter variation. Each node also runs a monitoring service to record relevant experimental data.

VI. Simulation Results

We observe that deploying additional Redis nodes allows us to connect additional clients. With 1 Redis node, we can properly handle up to 490 clients and with 8 nodes, up to 910 clients. Thus, Lamoth can allow a game to scale properly by reserving resources based on the projected load.

Outgoing bandwidth increases in a n^2-way as the number of players located inside the same area increase. Thus, a *flocking* behavior of a significant number of players triggered by a game event can quickly lead to an exhaustion of the available resources. Our experiments also reveal that adding additional Redis servers can aid in supporting the additional flow generated by many *flocking* players. Thus, should a major game event happening at some given location be expected, a MMOG game operator could reserve the appropriate resources and Lamoth would take care of the load redistribution. Our experiments also demonstrate that increasing the rate at which players perform actions can put an additional stress on the infrastructure and that Lamoth is able to handle such a stress provided that enough resources are provisionned.

VII. Conclusions and Future Work

We observed that the amount of messages that need to be transmitted in a MMOG game can often become a bottleneck. We proposed Lamoth, a platform that provides a software layer to handle message dissemination by making use of pub/sub nodes in the cloud containing ready-to-use pub/sub middleware. We integrated Lamoth into Mammoth. Our experiments reveal that Lamoth does allow a multiplayer game to scale towards impressive figures, confirming our assumption that the cloud can be successfully leveraged to support MMOGs even when some important bottleneck situations are expected.

As future work, we are working towards making Lamoth more dynamic by allowing cloud pub/sub instances to be spawned and despawned dynamically as needed. We would also like to perform simulations on real cloud infrastructure. We are also looking at how we could provide other MMOG game services as cloud services.

References

[1] J. Chen, B. Wu, M. Delap, B. Knutsson, H. Lu, and C. Amza, "Locality aware dynamic load management for massively multiplayer games," in *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, ser. PPoPP '05. New York, NY, USA: ACM, 2005, pp. 289–300.

[2] A. Bharambe, J. Pang, and S. Seshan, "Colyseus: a distributed architecture for online multiplayer games," Berkeley, CA, USA, 2006, p. 155;68.

[3] E. Lety, T. Turletti, and F. Baccelli, "Score: a scalable communication protocol for large-scale virtual environments," *Networking, IEEE/ACM Transactions on*, vol. 12, no. 2, pp. 247–260, 2004.

[4] M. Claypool, D. Finkel, A. Grant, and M. Solano, "Thin to win? network performance analysis of the onlive thin client game system," Piscataway, NJ, USA, 2012//, pp. 6 pp. –.

[5] M. Najaran and C. Krasic, "Scaling online games with adaptive interest management in the cloud," in *Network and Systems Support for Games (NetGames), 2010 9th Annual Workshop on*, 2010, pp. 1–6.

[6] J. Kienzle, C. Verbrugge, B. Kemme, A. Denault, and M. Hawker, "Mammoth: a massively multiplayer game research framework," in *Proceedings of the 4th International Conference on Foundations of Digital Games*. New York, USA: ACM, 2009, pp. 308–315.