Generalized Models of Network Architectures for Online Games

Guilherme B. Bedin
Hisham H. Muhammad
André Detsch
Marinho P. Barcellos

PIPCA - Programa Interdisciplinar de Pós-graduação em Computação Aplicada
Centro de Ciências Exatas e Tecnológicas
UNISINOS - Universidade do Vale do Rio do Sinos

{bedin,hisham,detsch, marinho}@exatas.unisinos.br

Abstract

Online games are network applications in which multiple players interact with each other (typically in real time) using computers or similar devices. The way computers are connected and application processes communicate impact on the application performance as well as network resources consumption. The effects of online gaming to the Internet traffic, and vice-versa, have only recently begun to be addressed in the literature.

This paper reviews existing interactive, online games. Based on their fundamental attributes, namely architecture and communication design decisions, it identifies generalized online game models. We envision their application in performance evaluation of current Internet games, the prediction of traffic they may generate in larger scenarios. We believe this will be the underlying model to research new, more scalable architectures for future Internet online games.

Keywords: *online games, multicast.*

1 Introduction

Online games are becoming one of the most important kinds of distributed application, and a main source of Internet traffic.

This paper addresses relevant communication aspects of online games, and as a result identifies generalized online game models based on their fundamental properties. All models are devised using the same set of key parameters, so that they can be compared under the same terms. Here we consider only games where participants interact with each other in real time during a single game session (e.g., first-person shooters, flight simulators, etc.)

2 Architecture

An online game can be described by its global state, that is, the current values assumed by game objects. At each player, the user display is updated many times per second, according to its local view of the global state. The global state is typically kept centralized at a single point or fully decentralized among players. Thus, online games can be generally classified according to their architecture in centralized or decentralized.

It is well-known that centralized, client/server architectures are poorly scalable; all messages need to go from clients to the server, be consolidated and then sent back to clients ([3]). Also, the processing performed at the server might become a bottleneck, as the number of players increases. Although combined network and processing delays might exceed the maximum “acceptable” values, the client/server is the predominant model in online gaming.

Games with fully decentralized architectures are also known as *peer-to-peer* or *serverless*: global state and decisions are shared among all participating computers. All nodes are equal: each one keeps a part of the the game global state and is responsible for a set of objects. Objects are distributed, but not necessarily replicated (objects like maps may be replicated, whereas there might be a single copy of each player-related objects). The game must use a synchronization mechanism to prevent consistency problems, like conflicting actions.

To prevent the above consistency problems, all events in all agents should be, at least ideally, globally ordered according to their wall-clock time. The real-time nature of interactive games, allied with current network technology, prevents decentralized games to employ a sophisticated event ordering mechanism. A solution involves a simpler distributed synchronization mechanism, such as the bucket synchronization scheme (see [4] for details) or specialized GPS hardware in each computer, so that clocks are synchronized with UTC global time with an accuracy of hundreds of nanoseconds.

So, the greatest advantage of the centralized architecture lies on its simplicity: all game events are inherently serialized by the server, allowing the game to easily order remote events in time (a single clock time exists, the server’s). However, the centralized architecture has long-known disadvantages too: there is a single point of failure, and a potential performance bottleneck at the server. In a decentralized architecture, the game logic and session control is distributed among all players in a way that permits a game to continue for some of the players if communication or node failures occur.

According to the design decision taken in regards to its architecture, online games will make use of one of two forms of multi-point communication. Multi-point can be achieved with multiple-unicast transmissions, though it scales poorly. Because of its simplicity and deployability, approaches based on multiple unicasts have been commonly used in online games. Broadcast allows a sender to efficiently transmit a single copy of each message to all nodes, but restricted to the same local-area network. IP multicast brings the benefits of broadcast without the cost of sending to all.

It is generally inefficient to fully replicate each game state entity in all players: players that are distant in a virtual world and cannot interact in any ways may have no interest in keeping copies of such game objects. Therefore, in a game with a large number of participants, a state update may not need to be propagated to all other players; a given player will be interested in receiving only certain data. The implications for client/server architectures is that the server will not need to send the same status update to all players; for decentralized architectures, each player will need to send (or to receive, depending on the viewpoint) only a subset of state updates (a set of data flows.)

3 Generalized Models

This section combines the attributes above defined, namely architecture and communication, to define four generalized models of online games: ***UC*** (multi-unicast, centralized); ***UD*** (multi-unicast, decentralized); ***MC*** (multicast, centralized); and ***MD*** (multicast, decentralized.)

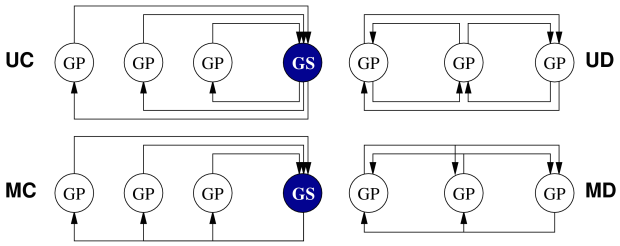


Figure 1: Four online game models.

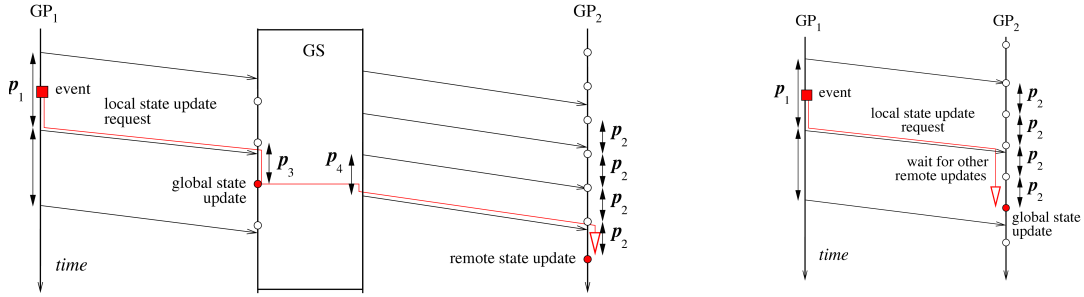
Figure 1 illustrates the models, which are discussed below. The centralized models, UC and MC, rely on a game server (denoted as **GS**) to order events, control game session, etc. In contrast, the decentralized models, namely UD and MD, are serverless, being the game logic distributed among all game players (**GP**s.)

In all models, there exists an application-level game engine that runs at every participant host and is responsible for interacting with users. The game engine displays to the user the local, current view of the global game state, periodically refreshing the console. In the centralized models, each player sends updates via unicast to all other players. The server, on its turn, periodically "consolidates" updates received from each player into a global state, and sends the result to all players, either via multiple unicast or multicast. In contrast, the decentralized models employ no server: each player directly sends and receives local state to/from all other players. Periodically, each player takes all local states received recently from other players and consolidates it into its view of the global state.

The proposed online game models are of *periodic* nature: all the processing performed by players and servers is done in periodic fashion. There are four time periods, which are: p_1, the period a GP sends out game state, either to a GS or to other GPs; p_2, the period a GP updates its local view of the global state; and for centralized models only, p_3, the period the GS consolidates game state received from GPs; p_4, the period a GS sends out consolidated state to GPs.

The four time periods are illustrated in Figure 2, which shows the state change dissemination from GP_1 to GP_2, for decentralized and centralized models (Figures 2 (a) and (b), respectively). In the decentralized case, (**i**) every p_1, GP_1 sends an update; (**ii**) every p_2, GP_2 processes a new local view of the global state. In the centralized case, (**i**) every p_1, GP_1 sends an update to GS; (**ii**) every p_3 the GS consolidates a new global state from received messages; (**iii**) every p_4, the GS will send updates to GPs; (**iv**) every p_2, GP_2 updates its local view of the global state, based on data received from the server.

The values of p_1 and p_4 represent sending periods (thus rates) of messages with updates. The shorter they are, the quicker the updates of a player tend to reach other players, but also the larger the bandwidth taken/required. In contrast, both p_2 and p_3 represent updating periods, for player and server, respectively. Their values affect the rate in which updates are communicated: a shorter p_2 will result in updates being processed and delivered more frequently to the user's display, at a higher processing cost; a shorter p_3 will allow the server to process more often collected state update requests received in messages. The new



(a) centralized (b) decentralized

Figure 2: State change dissemination from GP_1 to GP_2

global state computed by the server is useless unless it can be transmitted to players; therefore, $p_3 \geq p_4$. If $p_3 = p_4$, the server always computes new global state and then sends it to players; otherwise the server sends multiple times the same message, to add reliability through redundancy ("saturation").

4 Related Work on Online Games

One of the first online games to appear was Amaze ([1]), in the mid 80's. However, the first widely-popular games with network support came a decade later, usually following the MD model: combining a decentralized architecture (peer-to-peer) with link-level broadcast (usually IPX), limiting games to local area networks.

As online games migrated to the Internet, their networking systems moved towards a client/ server architecture based on UDP/IP (in our taxonomy, the UC model). In first-person action games (FPS), command latencies above 150ms are noticeable enough to the user to the point of affecting the interactive experience.

On the other hand, one may identify a different class of multiplayer games with very distinct network requirements. Strategy games (RTS) can cope with much higher latencies (up to 500ms [2]). Taking advantage of this, Age of Empires, for instance, uses a decentralized communication scheme built on top of UDP (following the UD model.)

While a promising alternative in terms of scalability for multiplayer games, multicast is still used only in the research field. MiMaze is a decentralized multicast-based game, thus fitting the MD model ([4]). It is novel in that it exploits IP Multicast to allow multiparty games to be played in the Internet.

IP multicast allows the efficient delivery of copies of the same information to a large set of receivers. However, this efficiency is limited by preference heterogeneity: when receivers range in their preferences for application data, the sender has to transmit all information to all receivers, so that most receivers will receive some useless information. This may be the case of certain online games. Related work on the scalability of distributed, interactive applications focuses on grouping, clustering or relevance filtering strategies. These are schemes that aim at reducing the amount of unwanted information that each receiver (in our case, player) will be delivered. To apply such grouping approaches with the current IP multicast architecture constitutes a challenge, since groups are subject to large setup overhead and long setup latency, including a long addressing procedure ([5]).

5 Concluding Remarks

This paper presented generalized models of communication architectures used in online games. As shown, these models, although abstract, reflect schemes used in existing online games. The main purpose of this modeling is to describe protocols and algorithms on a common ground, so they can be compared in performance evaluation studies.

These models can later be specialized in order to depict more precisely characteristics of a given category of online games. As future work, these models are being employed in simulations of multicast patterns for online games.

6 References

1. Eric Berglund, and David Cheriton, AMAZE: A Distributed Multi-player Game Program Using the Distributed V Kernel, IEEE Software, v.2, pp.248-253, May 1985.
2. Paul Bettner & Mark Terrano, 1500 Archers on 28.8: network Programming in Age of Empires and Beyond, In Proc. of Game Developer's Conference (GDC2001), San Jose, 20-24 March 2001.
3. Yahn Bernier, Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization, In Proc. of Game Developer's Conference (GDC2001), San Jose, 20-24 March 2001.
4. Laurent Gautier, C. Diot, Design and Evaluation of MiMaze: a Multi-Player Game on the Internet, In Proc. of IEEE Multimedia Systems Conf., Austin, June 28 - July 1st, 1998.
5. Brian Neil Levine, Jon Crowcroft, Christophe Diot, J. J. Garcia-Luna-Aceves, James F. Kurose, Consideration of Receiver Interest for IP Multicast Delivery, In Proc. of IEEE INFOCOM 2000, Tel-Aviv, 26-30 March 2000.