Generalized Two-Tier Relevance Filtering of Computer Game Update Events

Lars Aarhus
Norwegian Computing Center
Postboks 114 Blindern
NO-0314 Oslo, Norway
aarhus@nr.no

Knut Holmqvist
Norwegian Computing Center
Postboks 114 Blindern
NO-0314 Oslo, Norway
holmqvist@nr.no

Martin Kirkengen
Norwegian Computing Center
Postboks 114 Blindern
NO-0314 Oslo, Norway
martin@nr.no

ABSTRACT

In this work-in-progress paper we present a relevance filtering scheme for a two-tier server architecture optimized for massive multiplayer online games. We distinguish between *interest management* of server tier game state and *bandwidth adaptation* of concentrator tier client link thresholds, making the concentrator tier totally application independent. An initial prototype has been implemented, demonstrating significant reductions in update event rate without loss of playability.

1. BACKGROUND

Multiplayer computer games are increasingly popular in the public. For a long time they were limited to non-distributed desktop games, with up to four players in the same room, but in recent years massive multiplayer online games have caught on. No longer restricted to military virtual environments in dedicated networks, e.g., SIMNET and NPSNET [6], distributed real-time computer games communicating through the Internet is now a fast growing field, e.g. Ultima Online, Everquest and Anarchy Online. All these games face the same problem of *scalability* - how to accommodate as many users as possible within the same game [3].

1.1 Scalability Challenges

Approaches for scalable server architectures typically involve distributing the server across several entities. *Distributed* server architectures spread the server load on several machines working in parallel, usually with separated computing tasks. We use the term *two-tiered* for server architectures where network communication is limited to a dedicated concentrator layer. The architecture outlined in this article is both distributed and two-tiered.

The client–server networking also poses the challenges of efficient bandwidth usage between server and clients, typically involving a combination of different techniques.

- *Data Compression* of transmitted game updates.
- *Relevance Filtering* by transmitting only a subset of game updates based on the interests of each client.
- *Multicasting* of game updates by group communicating identical messages to several clients.

NetGames 2002, April 16-17, 2002, Braunschweig, Germany.

Of these techniques, only relevance filtering will be discussed in this article. Our distributed, two-tiered architecture places certain demands on the relevance filtering, which again provides interesting new functionalities.

1.2 GISA

GISA (Generic Internet Scalable Architecture) is a 3-year project (2001–2003) aimed at developing a middleware architecture able to support massive multiplayer online games. The strong scalability requirements have lead to the development of a two-tiered server platform.

The main focus of GISA in its first year is on developing *generalized methods for relevance filtering* of game updates between players.

Gaming networks also face the problem of latency (end-to-end delay) between the players. The design presented in this article is optimized to reduce latency. It also includes methods for latency masquerading, using *dead reckoning* to predict local client states. We have developed a method for *time synchronization* of clients using TCP, to allow timestamping of events. However, these methods are not the topic of this article.

2. ARCHITECTURE

The GISA is a scalable client–server architecture in which the server side is organized in two tiers: the *server tier*, which contains the game state logic, and the *concentrator tier*, whose task it is to perform connection and bandwidth management.

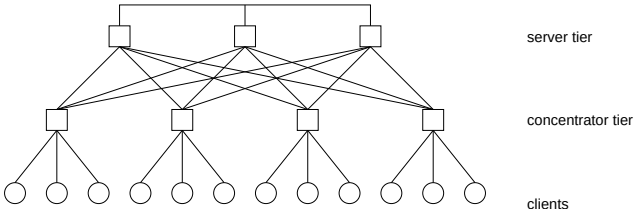

Figure 1: Architecture tiers.

The architecture makes two important assumptions about the characteristics of the network infrastructure:

- The network between servers and concentrators is characterized by high bandwidth, low latency, and low packet loss.
- The network between concentrators and clients is in the worst case characterized by low bandwidth, high latency, and significant packet loss.

The architecture was selected to optimize scalability, while allowing the maximum interaction between all players.

2.1 The Server Tier

The server tier is a distributed system that supports distribution of the game state over a number of physical machines in order to handle the load. Each server in the server tier communicate with all the other servers and all the concentrators.

A server is an event based system; it receives events from various sources, and according to its internal state (called the *world model*) it generates new events that are routed to the appropriate recipients. Since the server tier distributes the game state over several servers, events generated by the world model of one server may need to be passed to the world model of another server.

The rationale for the event-based design is that the server tier must be able to handle massively parallel real-time games. The event-based system allows asynchronous communication within the server tier, and is more robust to client-concentrator latency. A turn-based system would not be able to handle these demands adequately, as waiting for one client will reduce playability for all the others. In our system, only players in direct interaction with the high-latency client will be affected.

2.2 The Concentrator Tier

The concentrator tier consists of a set of concentrators, each of which is connected to all the servers and a large number of clients. Events received from clients are routed to the servers in the server tier. Concentrator functions include

- connection and bandwidth management
- event routing between clients and servers
- duplication of events to clients
- filtering of events to clients

A concentrator adapts to the characteristics of the network that connects its clients, so that even players using clients on poor connections can get a subjectively good perception of the game.

Clients connect to concentrators based on network topology, not their position in the game world. The concentrators are constructed to be application independent, allowing multiple games to use the same concentrators, even simultaneously.

2.3 The Client

The client communicates with the user through a graphical user interface and maintains a partial model of the game state. This model is not necessarily accurate; the game state maintained by the server tier is by definition correct, and the client state may deviate slightly from this state.

The client game state is updated by events sent from the server tier via the concentrator tier, and player action events are sent from the client to the server tier. The client is robust towards varying network conditions, but the gaming experience may suffer.

Also, the correct operation of the GISA middleware architecture does not depend on the correct operation of the client; the middleware is not affected if the client should crash or get disconnected from the concentrator. While in some games, like real time strategy games, this will lead to a breakdown of the gaming experience, it matters less in a persistent role-playing game.

3. RELEVANCE FILTERING

The purpose of relevance filtering is to ensure that each client receives as much information as possible about that subset of the game world that is relevant to the player. Two principally different definitions of relevance can be discerned. One is based on the *server-side game state*, and is based on the players interest in the data. The other considers the *transmission process*, where data can become more or less relevant depending on transmission delays or previous transmission priorities or failures. The following examples illustrate this:

1. A rock artist enters stage. Everyone in the room will immediately know. People a block away will not know at all, and in general, do not care. Relevance is defined by what is visible, audible or interesting, based on the game state.

2. Fast movements in combat. During fast combat-like interaction, the important information is the exact, instantaneous position. If a packet describing this information has not been transmitted by the time a new position is estimated, the previous packet is obsolete and should not be transmitted.

Decisions based on game state have to be made on the server, as the game rules and state are not known to the concentrator. But the server has no knowledge of the network conditions a given client is experiencing at a given time, so decisions based on the transmission process have to be made by the concentrator. This leads to the following, important definitions:

- *Interest management* - server-side relevance filtering based on game state.

- *Bandwidth adaptation* - concentrator-side relevance filtering based on networking conditions.

3.1 Related Work

Interest management for update events has a long history, in particular within the field of large-scale virtual environments and distributed simulations. Various schemes have been suggested in conjunction with the High-Level Architecture (HLA) of US military simulations. Common to most systems are single-tier filtering, static IP multicast event distribution and an inherent assumption of reasonably predictable network conditions. See [5] for a survey.

In [1] a three-tiered approach for interest management is suggested, utilizing dynamic IP multicast group assignment. It operates with dynamically changing regions, and two levels of fidelity and update rate, in addition to a protocol dependent third tier.

In [8] the focus is on partitioning and distributing the shared state in distributed simulation. A hierarchical, multi-level dynamic interest management scheme is suggested, using “spheres of influence” for load balancing of shared state processing.

In [4], filtering and addressing, two techniques to scope delivery of content to interested receivers in IP multicast are contrasted. It is shown that addressing is preferred to filtering given that multiple multicast groups are easy to create and manage, which is not the situation in the current Internet architecture. Filtering in this context is broadcasting of data to all receivers using a common multicast group followed by receiver filtering prior to passing data to the application.

Bandwidth adaptation has to some extent been approached in two different ways. In first-person-shooters, UDP is used for overridable messages like movement and TCP for critical messages like killed players or demolished structures. Some military applications with QoS guarantees, allow graded interest definitions with a runtime threshold adaptation [6,7]. We do not know of any attempts to *combine* these two approaches, or to perform the threshold adaptation with *unpredictable bandwidth*.

3.2 Interest Management Generalization

Interest management is often referred to as "area of interest management". This is due to the strong connection between spatial representation and relevance. Most sensory mechanisms and interactions are based on objects being close to the player in space.

However, many intuitive distinctions of "interesting" are not compatible with fixed "areas", the way these are traditionally defined.

First, different interactions have different ranges, so one should consider a different "area" for each type of interaction. Some of these "areas" can be more general parameter spaces, like a radio frequency band [7]. Similarly, a person you know will stand out in a crowd, even if you notice no one else, and hearing your name spoken at a party will immediately draw your interest even though you have heard nothing else of the conversation. Also, as the interaction is centered on the player, every player will have his own notion on what is the optimal area definition.

This concept is elaborated and generalized in [2], which introduces a spatial model of interaction for objects (e.g. players), including key abstractions such as aura, focus and nimbus.

In addition to the potential interest because of the interaction, what is perceived through the interaction is of graded importance. While you may ignore a person walking slowly, you may notice the same person in the same position making a fast change of direction. This change will also make any client side prediction being significantly poorer, as the client will expect the old behavior to continue until told otherwise. Thus some state changes are more important than others, even though the interaction is the same.

Because of this multitude of simultaneous "area" and "importance" definitions, we have chosen not to limit the interest management to the traditional concept of "events in the same area". A set of interest metadata is used, describing the criteria for judging the relevance of a certain event or group of events for a given player or a group of players, similar to the approach used in the HLA [7].

Another important aspect that is considered in our approach is that of "graceful degradation". If the standard interest classification leads to too much total data being marked as interesting, because a room is overcrowded, or if the available bandwidth suddenly drops, this should not lead to critical errors in the client game state. This includes protecting clients from having some data consistently starved, by increasing the priority of messages replacing discarded messages.

3.3 Selected Approach

Based on the above considerations the relevance filtering is divided into the following separate functionalities:

1. *Interest indication and subscription.* The process where the server defines what types of events a certain client is interested in. This is an arbitrary, game specific process, and can be linked to the collision detection, or changes in certain parameters. The results of this process are transmitted to the concentrator as subscription messages.

2. *Metadata marking of events.* The server side marking of events must obviously match the marking used in the subscription process. Events are marked as referring to a given object, but in addition they are to be filtered by an "interest-defining-object" (which may be the same as the handling object), and they are given an objective "importance". The event is then transmitted to the concentrator for filtering.

3. *Event filtering due to subscriptions.* On the concentrator, events are passed through the subscription list. In this step, the importance of the event must be determined from the combined values of the event importance and the client's interest. This step may also easily be modified to accommodate multicast groups.

4. *Individual bandwidth adaptation.* Finally, the concentrator transmits events as fast as possible, always picking the most important event to be sent next. If a new event arrives that refers to the same information as a previous unsent event, the old event is discarded, and the priority of the new event is increased, up to a maximum. Different algorithms, e.g. logarithmic increase, may be used for the priority calculation.

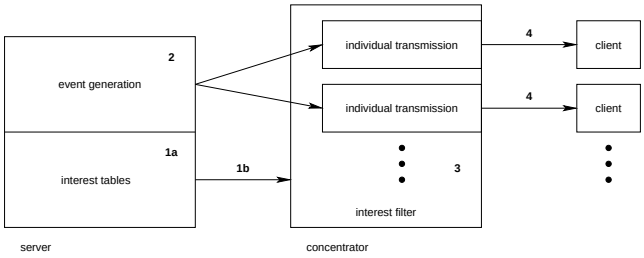

Figure 2: Relevance filtering functionalities.

Two important issues must be noted. First, we require the changes in potential relevance to occur much less frequently than the actual number of events. This is necessary to justify the proposed architecture.

Second, the concentrator side functionality can be made totally *application independent*. The concentrator only regards the abstract "interest-defining-objects", the client connections and the "importance", the rest being totally encapsulated in the event packets. All game specific logic is restricted to the server. This also allows *runtime updating of the server* without affecting the client connections.

4. MULTI-TAG: AN IMPLEMENTATION

In order to demonstrate the GISA middleware architecture we developed a net-based version of the children's game Tag, only with multiple "tagged" players, hence Multi-Tag. The "tag" is transferred from one player to the next in collisions. The game was selected because it is intuitive, with simple objectives and interface, while being very vulnerable to latency or game state inconsistencies. The game is implemented in Java, using only TCP connections so far.

The user interface of the game has two panels. The largest panel, called the "main view", represents a relatively small area around the player on the playground. A smaller panel, called the "minimap" shows the whole playground, with an indicator of where in the playground the "main view" is located.

Multi-Tag let us demonstrate several aspects of interest management. "Standard" area of interest management is used by transmitting all events occurring inside the "main view", and only important messages outside the "main view". Normal movement (minor changes) are marked as unimportant, and collisions (major changes) are sufficiently important to be transmitted. Other relevance filtering concepts are used to mark all the tagged players as interesting to everyone else, without even changing the interest tables as players are tagged/un-tagged. The "minimap" gives a good visualization of the relevance filtering, as un-tagged players outside the "main view" move jerkily, or not at all, while the tagged players move in real time.

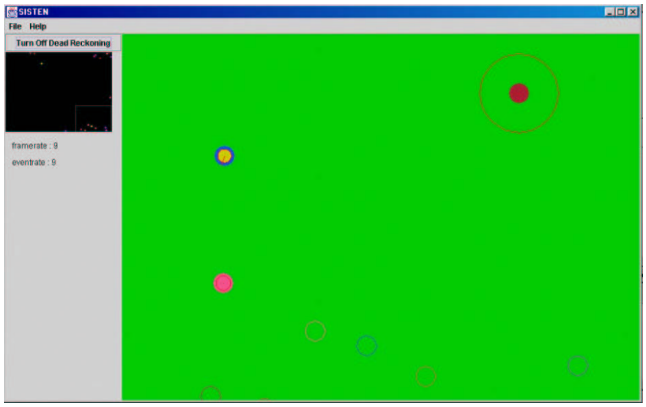

Figure 3: Multi-Tag user interface and playground.

From the clients, each player keystroke is transmitted, with the time and position when it was pressed. "I-am-alive" events are regularly sent to avoid uncertainty in whether packets have been lost. The server interpolates between these discrete positions and sends a confirmed position back to the players, or a new position and speed if a collision has occurred.

For demonstrational purposes the different interest management schemes can be turned on and off.

4.1 Results

So far, significant reductions in update event rate (up to 90% under some conditions) have been demonstrated, without introducing any observable extra latency or reducing playability of Multi-Tag. The effect of the relevance filtering will obviously depend on the size of the "area of interest"relative to the full game world, available bandwidth, and the tuning of the different interest definitions.

We do not include proper statistics as all testing has not yet been completed. The proposed relevance filtering approach has been tested in lab environments with server and concentrator tier on different physical machines, as well as clients connected over fast Ethernet links. However, the real interest is to observe the effect in home environments with low-bandwidth, high-latency clients e.g. connected through traditional modems.

Also, any reduction numbers are game dependent and difficult to compare. A 60% reduction in update event rate for Multi-Tag as typically observed, does not reduce the playability of that particular game. Whereas a lower, e.g. 40%, reduction for another type of game might do, since the playability threshold is lower.

5. DISCUSSION

This work-in-progress paper presented a *generalized two-tier server approach for relevance filtering* of computer game update events. Server scalability was the main motivation for the architecture, but the constraints posed by the architecture helped structuring the relevance filtering properly, and led to the distinction between *interest management* and *bandwidth adaptation*.

The architectural separation of interest management and bandwidth adaptation offers easy balancing between the two concerns, as well as concentrator independence when the application game logic is modified. Changing the game on the server does not influence the concentrator, which is advantageous for real-world game providers.

Our approach is two-tiered, but the tiering is server architectural, not protocol focused as in [1]. However, support for multicast can easily be included in the concentrator tier by assigning different subscription groups to multicast addresses at protocol layer. The need for shared state distribution and processing is reduced by dead reckoning methods at the clients and our choice that game state is residing on server.

Since our relevance filtering is performed in the two-tier server, and not in the receivers (clients), the findings in [4] concerning resource inefficiency of filtering in network and clients are less valid. On the other hand, sufficient server and concentrator processing resources are more critical because of necessary subscription management.

The time synchronization of our approach tries to account for the varying latency between entities (clients). The event-based server ensures robustness to communication failures with clients, only the players interacting directly with the failing client will be affected.

The total design aims at relatively massive number of players, in the hundreds or thousands, but we have only been able to perform a limited testing so far.

In the next two years the relevance filtering subscription methods will be further enhanced and tested in the GISA project. More real-world experience and concrete update event reduction numbers will be gained as the approach will be evaluated in other demonstrator applications besides Multi-Tag.

6. ACKNOWLEDGMENTS

Thanks to Syncrotec A/S and Norwegian Research Council for funding the GISA project.

7. ADDITIONAL AUTHORS

Additional authors: Thor Kristoffersen (Norwegian Computing Center).

8. REFERENCES

[1] H. Abrams, K. Watsen, and M. Zyda. Three-tiered interest management for large-scale virtual environments. In *Proceedings of 1998 ACM Symposium on Virtual Reality Software and Technology (VRST'98)*, Nov. 1998.

[2] S. Benford, J. Bowers, L. E. Fahlen, and C. Greenhalgh. Managing mutual awareness in collaborative virtual environments. In *Proceedings of 1994 ACM Symposium on Virtual Reality Software and Technology (VRST'94)*, Aug. 1994.

[3] C. Fitch. Cyberspace in the 21st century: Scalability with a big 's'. Gamasutra.com, Feb. 2001.

[4] B. N. Levine, J. Crowcroft, C. Diot, J. Garcia-Luna-Aceves, and J. F. Kurose. Consideration of receiver interest for ip multicast delivery. In *Proceedings of 2000 IEEE Infocom*, Mar. 2000.

[5] K. L. Morse. Interest management in large-scale distributed applications. Technical report, Department of Information & Computer Science, University of California, Irvine, 1996.

[6] S. Singhal and M. Zyda. *Networked Virtual Environments*. Addison-Wesley, 1999.

[7] R. Smith. Cutting-edge techniques for modeling and simulation. Tutorial, Game Developers Conference 2001, Mar. 2001.

[8] G. Theodoropoulos and B. Logan. An approach to interest management and dynamic load balancing in distributed simulation. In *Proceedings of the 2001 European Simulation Interoperability Workshop (ESIW'01)*, pages 565–571, June 2001.