Aspects of Networking in Multiplayer Computer Games Review

This paper was publish in The Electronic Library Journal in 2002. This journal is primarily geared towards applications of web based libraries. Interestingly enough, the issue in which this paper appeared seemed to have an online computer game theme.

Online multiplayer games were a relatively new medium at the time, but their prevalence and complexity expanded exponentially during and following the time of this paper.

The abstract claims that multiplayer computer games (MCGs) are at the forefront of utilizing network possibilities. As a result, they offer an overview of four aspects that affect networking in MCGs. These are—network resources such as bandwidth and latency set boundaries in which the MCG must operate, communication, data, and control architecture is encompassed by distribution concepts, scalability provides MCGs the ability to adapt to resource changes by parametrization, and security issues against cheaters and vandalism.

MCGs belong to shared-space technologies, along side military simulations utilizing virtual reality. In the 1980s the US began developing military applications utilizing the protocol Distributed Interactive Simulation (DIS). This protocol dealt closely with the properties of military units and vehicles. Following DIS the military concentrated on developing a high level architecture (HLA) to provide a general service for data exchange. HLA has the possibility of being used in non-military applications such as computer games, which has yet to happen but there has been successful cooperation between military and entertainment industries. In the following years several other virtual reality architectures had been developed for distributed real time applications, MCGs not being one of them.

The problems faced by MCGs had been realized by the entertainment industry, but not much work had been done on the topic in the scientific field. This paper intends to identify relevant aspects of networking that affects MCGs.

There are three limitations facing network resources; bandwidth, latency, and host processing power for handling network traffic. Bandwidth is essentially the measure of data transmitted per unit of time. An issue facing bandwidth requirements is the number and distribution of users. With multiple users, sending unicast packets is quite an inefficient use of bandwidth. This is where multicast comes in, allowing one source to transmit a single stream of packets to a group of multiple receivers. Latency is the measure of time it take for data to get from the source to the destination. Jitter is when there are significant variations in latency, and has a noticeable effect on interactive applications. While latency can never be completely eliminated, a certain level of it can be tolerable. For MCGs this level is between .1 and 1 second. Although the level at which it becomes apparent to the user varies greatly among different genres of games. As for processing power, handling network traffic creates computational strain on the host computer that is often overlooked.

Since there is not much that can be done about these issues, the problem must be solved by choosing architectures for communication, data, and control. There are several models of communication architectures. The most basic of these is one computer, with a split screen. Next is a peer-to-peer architecture in which all computers are directly connected and multicast to one another. Then there is the client server architecture in which one node acts as the server, and all communication is handled by it. Finally there is the server-network or server pool architecture, which is a combination of the peer-to-peer and client server architectures. Essentially it is a peer-to-peer network of servers which each have their own client server architecture. This provides an improved possibility of scaling because it reduces the requirements of each individual server.

As for data and control architecture, a decision must be made on the tradeoff between consistency and responsiveness. If consistency is a top priority, than a two-way relay which sends local control messages to the network and data back to the node, is the best bet. If responsiveness is the top priority, then a short-circuit relay in which local control messages are sent to the network as well as passed back to the node, is the best bet. As for MCGs, responsiveness is important and thus a distributed or replicated architecture that allows for short-circuit relay is best. Distributed means each node holds a subset of the data, and replicated means all nodes hold a copy of the same data.

Architectures on their own are not always enough to reduce resource requirements, and further measures must be taken such as compensatory techniques to reduce traffic sent between nodes. One such technique is message compression and aggregation. This compresses the messages consuming extra computational power but saving bandwidth, while aggregation reduces the frequency of transmissions by merging multiple messages together. Another technique is interest management, in which nodes express interest to only a subset of relevant information saving bandwidth. The data of interest is called an aura, so when two player's auras overlap they see each others actions. This can be expanded even further by dividing the aura into a focus and a nimbus, representing the observer's perception and the observed perceptivity, respectively. Finally there is the technique of dead reckoning, where one's position is estimated based on a known starting point and velocity. This can also be applied to predicting the data of other nodes, allowing for longer message intervals and a solution to the issue of latency at the cost of data consistency.

Scalability is an extremely important aspect of MCGs. Sequential computation limits the potential speedup obtained by parallel computation, but the two can be separated into two execution time formulas. Amdahl's law for a fixed problem setting is an optimal formula combining the two.

Assuming clients have asynchronous messaging, a worst case scenario would be if all nodes attempted to communicate at once. This limits scalability due to communication capacity, which is different for each deployment. So in order to overcome this limitation, nodes cannot always be aware of all other nodes, thus a sublinear communication is key. To achieve this, the communication between servers

must be limited. This is where the previously mentioned compensatory techniques come in handy. Messages can be compressed and aggregated, but this is usually not enough. A better solution is to use interest management to reduce the amount of messages sent in conjunction with dead reckoning to approximate suppressed messages.

Security is a big issue with online games, as they are open to the public and frequently exploited. This topic has not been covered before in related research [at the time of this paper]. On the other hand, military simulations aren't concerned with this seeing as it is strictly private and not open to the public.

The two main goals of security for online games are considered to be protecting sensitive information and providing a fair gameplay experience. One of the common methods of cheating is packet and traffic tampering. This is commonly used in FPS games, in such ways as monitoring the position of opponents and calculating the correct trajectory needed to hit before firing. Another method is to simply suppress packets containing damage to the cheater, rendering them invulnerable. Finally, a packet can be sent repeatedly turning a single shot weapon into a rapid firing one. One way to prevent these cheats is using MD5 checksums, however cheaters can reverse engineer the checksum and can also use the replay attack. A better prevention is to encrypt command packets, and include randomly generated numbers to differentiate identical packets, or even add a random amount of junk data eliminating the possibility of analyzing contents based on size.

Another common method of cheating is to use a cracked client that allows access to hidden game data. An example of this implementation is seeing through walls in a FPS game. This problem is derived from the software and cannot be prevented solely with networking. The sensitive data should be encoded and difficult to find in memory, but is still susceptible to determined hackers. A solution in a centralized architecture is for the server to detect whether a command issued by a client is actually aware of the object it is targeting.

A final common way of cheating is based on design defects. Hackers can detect loop holes in the design of a game and exploit them gaining unfair advantages. The best way to deal with this is to simply alter the design rendering the exploit impossible. Also, due to the network environment unexpected behavior may occur in the presence of high latency or when the server is under a DOS attack.

The abstract essentially lists the four topics about multiplayer computer games that the paper will address. It did in fact address these points, but it gave the implicit impression that actual solutions would be provided, which is not the case.

The transitions between subsections was done very well, but didn't exist between the 6 primary sections.

All of the talk about military simulation and such seemed very out of place, as it was barely touched

upon and does not have all that much to do with multiplayer computer games except a few similarities, and wasn't useful information to know with respect to the topic and goals of the paper.

While the authors defined most of the key terms, it was very off balanced. For example there is an entire paragraph explaining what bandwidth is, while the term relay (in respect to data and control architecture) is never defined.

It is not very clear as to why the first graph (fig 1) is needed, or how to even interpret it. On the other hand, all of the other diagrams proved very helpful, with clear explanations in the text as well as subscripts under each figure re-explaining so you don't have to jump around the paper to figure out what the figure really means.

The example of communication capacity in client/server architecture in which they actually plugged numbers into the provided formulas was confusing at first, but proved helpful in the end.

This paper was written in a very high level context. There were no detailed descriptions of protocols, packet headers, or anything of the sort. Thus the formulas provided for serial and parallel computation stood out like a sore thumb due to the level of previous knowledge required to actually understand and appreciate them.

This paper is very odd in the sense that it appeared in a journal that primarily covers applications pertaining to web libraries. This paper does not provide any applications, nor does it provide instructions on how to implement one, and clearly has nothing to do with libraries of any sort.

From the view of someone outside the field with a limited knowledge of computer science and an interest in the basic architecture of multiplayer games, this paper could be an interesting read. As far as actual usefulness in the academic or industrial world, it does not provide any new information, or detailed enough information for an implementation. Most if not all people with a strong background in computer science could come up with the few basic solutions they provided. The paper simply states various obvious problems faced by multiplayer games, followed by brief high level descriptions of simple architectures that could solve some of the issues.