Paper Analyses

Paper: "Aspects of Networking in Multiplayer Computer Games"

Multiplayer computer games (MCGs) have become very popular, especially in the last five or so years. Major companies, like Blizzard, got online gaming sites (Blizzard has Battle.net) which offer multiplayer-gameplay to their users. There is a massive demand for multiplayer-gameplay, and it is becoming more and more popular. In addition to these, mobile entertainment has skyrocketed, creating an even higher demand. This implies that technical and practical challenges related to MCGs, will be around for awhile.

Solutions from related research have diverged from the problems of MCGs. The prior proposed solutions have been for military simulations, virtual reality systems, and computer supported cooperative work. After 1992, the military research were concentrating on developing a HLA (high-level architecture) that would provide a general architecture and services for distributed data exchange. Even though it was designed for military purposes, it could also be used in other areas, such as computer games.

MCGs represent a fourth class even though they have same qualities as DVEs, CVEs and DIS: real-time computer applications. Regarding resources, distributed simulations face three limitations: bandwidth, latency and computational power (in terms of handling network traffic). Computational power did not occur to me at the beginning, but as the paper states, as the number of users reaches a number of user which has six digits, it is going to be a lot of traffic that needs handling. Bandwidth requirements are also related to the number users, in addition to the distribution and transmission technique. The authors provide some information about unicast, multicast, and broadcast. Not surprisingly, unicast is not a good alternative since it would require a lot of redundant messages (as there are a lot of receivers). It is impossible to get rid of latency. It is, however, possible to make it as low as possible. Of course, different games/applications will have very different requirements, and should be taken into consideration when making the game/application. Real-time strategy games could accept up to 500ms, but a first-person shooter (FPS) game would suffer from this. The paper says that FPSs require latency closer the 100ms, but gamers today are not pleased with more latency than the low order of tens ms.

It is not much that can be done with these limitations, so the solutions have to come at a higher level. In other words, we need to find out which architectures for communication, data, and control that gives us the best performance/solution.

Regarding communication architecture, there are four different degrees of deployment: (1) split screen on a single computer. (2) peer-to-peer architecture, (3) client/server architecture, and (4) server-network architecture. They all got their, rather obvious, advantages (like split screen will not suffer from latency caused by the network). Which to choose, will depend on the number and distribution of users, scalability, and server performance and availability. In real life development, one would also have to consider the resources at hand: how much time do we got, enough resources, interest in the product from the public (falls kind of into scalability), and so on.

In data and control architectures, there are two key attributes: consistency and responsiveness. Both cannot be achieved at the same time as high consistency requires tightly coupled remote nodes, and high responsiveness requires them to be loosely coupled. The high responsiveness attribute includes more computations at the nodes so that the bandwidth and latency do not play such a significant role.

On the other hand, high consistency requires high bandwidth, low latency, and a small number of remote nodes. On to the three architectures: centralized, distributed, and replicated. Centralized might seem as a good architecture where all the data is stored at one node. This provides consistency, but is lacks the responsiveness required by MCGs. Distributed and replicated architectures are more suited for this as they provide the responsiveness needed. The difference between these two are not that big: while a node holds a subset of the data in distributed architectures, all nodes holds all the data in replicated architectures. Distributed architectures are more suitable when the player behavior is unpredictable and the control commands can only have one source. A game with non-player characters (NPCs) and other predictable instances that do not have the need of sending frequent control messages, replicated architectures are the way to go.

As mentioned earlier, we cannot (or maybe in a very little degree) affect the network resource limitations, but it is possible to reduce these requirements. Three interesting approaches were presented. First, it is possible to compress and/or aggregate messages. In this way, the number of messages sent is reduced. The downside is that it requires more computational power. The trade-off has to be considered in the making of the system. A second approach is interest management. This basically means that nodes express what subset of information they are interested in. For example, if you are playing Blizzard's World of Warcraft, you do not need information about the users that are in different location than you. This seems like a pretty obvious thing, and I wonder why this was not implemented from the start. Final approach: dead reckoning which is the prediction of an object position based on its current position and velocity. This were not a new concept, we have even seen it in the "MPEG"-paper from 1991.

When talking about scalability, serial and parallel execution is discussed. They state that the theoretical speedup gained by parallel execution is 1/n. This is the potentially and ideally result, but in reality the speedup is much less. From reading the paper, it seems like they mean that 1/n is actually something they can obtain, which they cannot. As I guess they knew this, they should have clarified it.

The capacity requirement changes a lot depending on what architecture that was chosen. For example, peer-to-peer has n if multicast is used, and n^2 if unicast I used. This could probably make a big difference in performance.

Ignoring *interest management* would make the game unscalable. It is therefore important that a client cannot be aware of all the other clients all the time. This would result in a lot of messages.

The last part of the paper addresses security. The whole section is imprecise, and only gives a very general overview of some problems. It seems like they just threw it in the paper to cover it. It would more interesting to see good solutions to the security problems, and a paper only addressing security in MCGs would be something I would like to read.

Overall, the paper presents some good ideas and aspects to MCGs. It gives the developers something to think about. The paper was not hard to read and not swamped in formulas. It contained some figures which made it easy to follow the concepts described. Figure 3 is a good example where the reader is able to extract the information quickly. The introduction was OK-- they presented their goal, motivation, discussed prior solutions, some background information and the layout of the rest of the paper.

Abbreviations
CSCW – computer supported collaborative work
CVE – collaborative virtual environment
DIS – distributed interactive simulation
DVE – distributed virtual environment
HLA – high-level architecture
MUD – multiuser dungeons
NPC – non-player character
VR – virtual reality