Chapter 3 Related Work

The analysis of a next generation MMOG communication model is described in this chapter. Also, some significant middleware technologies aim to support MMOG application will be introduced in this chapter; most of them are commercial products.

3.1 Common Communication model in Online Games

Communication architecture is an important part of the design of a MMOG platform. Here is a survey of three typical kinds, namely, *peer-to-peer*, *client-server* and *proxy/gateway-based* architecture, focusing mainly on the issues of suitability and scalability that are specifically relevant to the proposed platform.

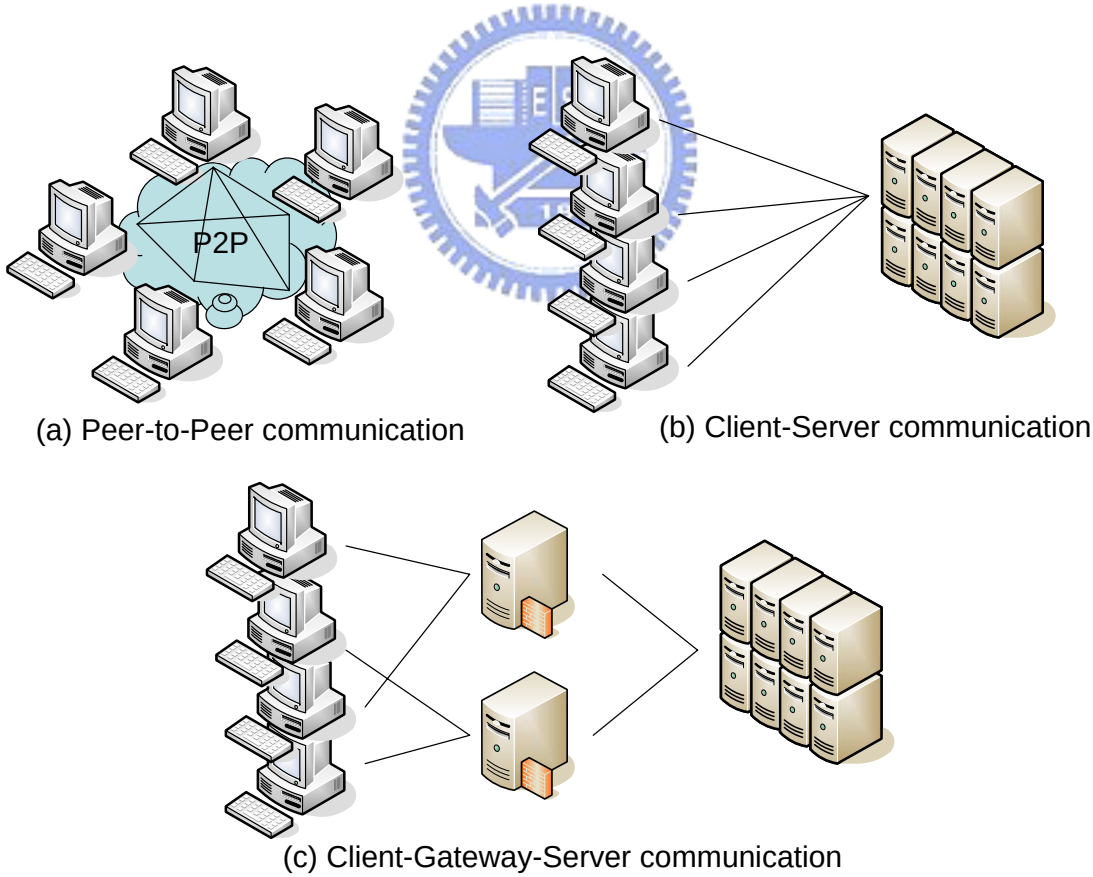

Figure 3-1. Three typical communication architecture of MMOG

3.1.1 Peer-to-Peer Architecture

Even though peer-to-peer-based architecture (as shown in Fig. 3-1 (a)) has proved its use in Age of the Empire [4], it obviously becomes very difficult to scale as the numbers of players increase. Reducing the high communication cost while maintaining consistency and player collaboration is a considerable task. Since it places the burden all on the player's computer and because of the limited network resources, peer-to-peer architecture is hard to scale. In addition, much of the unverified game the play information is in client, which leaves very difficult security issues unsolved. In recent years, Muller et al. [23][24][32] and Ferretti [46] try to purpose a mirrored game server with P2P technologies to solve the problems that is always critical problems to a commercial MMOG, such as consistency, and fairness. However, it seems that this type of architecture is far better suited to the small rather than the massive multiplayer game world.

3.1.2 Client-Server Architecture

Today's online game market, particularly the MMOG sector, is dominated by traditional client-server-based architecture. High-volume MMOGs often use clusters of servers to support their architecture (see Fig. 3-1(b)), such as Ultima Online [16], War of Warcraft [6], or Lineage [37], so client-server enjoys the advantage of centralized system management and scalability using an applicable server cluster load balance mechanism. The problem is that it does not scale well on the Internet while some clients have long latency to the server. In addition, the limited bandwidth at the entrance of server clusters can constitute a bottleneck.

3.1.3 Proxy/Gateway-based Architecture

Fig. 3-1(c) illustrates the proxy-based architecture. This type solves the Internet bandwidth and response-time problems while local proxies are close to clients. This architecture is also well-suited to the MMOG environment and clients. The clients treat the proxy as server and it shields the server from the clients. The latency between the proxy and server can be optimized by delegating the appropriate Internet Service Providers (ISPs) to deploy the remote proxies. Mauve et al. [30][32]indicate that part of the server functions can be assigned to the proxy, which means that it can help the server, for example to distribute loading or prevent cheating. The proxy can also use specific hardware to improve scalability [11]. Furthermore, the proxy can be seen as an extension to the server with a flexibility of design and deployment, and the information coming from the proxy can be trusted.

3.2 Next Generation MMOG architecture

In the previous chapter, we highlighted many problems (development, service, security…etc) of concern to the vendor. Everyone agrees with the idea of using middleware as software infrastructure to build an MMOG. With it, programmers can manage the complexity and heterogeneity of distributed computing environments. It is also an important integration tool, as an increasing number of companies – as a result of mergers, acquisitions, and infrastructure upgrades - try to assimilate multiple systems and applications[1][30]. Because of these issues, a virtual world platform solution with distributed technologies helps the vendor build a MMOG quickly and have no worries about network transmission and collaboration between servers or clients

Actually, the idea of using middleware to support an MMOG is not new. Many companies or research devoted to building frameworks to support MMOG development. The technologies involved are generally based on a distributed system and n-tier architecture, which are illustrated in Figure 3-2 as 4-tiered, comprising client, proxy/gateway, cell server, and database. Client is the main program of the graphics/user interface/communication controlled by the gamers. Proxy/gateway acts as a security defender of the cell servers. It also cooperates with the portal or billing system. Cell server is where the virtual world is located, executed, and maintained. The server receives control commands from the virtual world players, verifies and updates the states in cell server and then sends them to the players. To ensure their continued existence in the virtual world, the player states in the cell server cluster are periodically stored in persistent storage (i.e., in the database). This architecture is suitable for billing and anti-hacking in MMOG ecology, as well as in the enterprise computing environment and thus most commercial products follow use it. Although Peer-to-Peer (P2P) technologies, such as Knutsson[28], can benefit from the computing power and bandwidth shared by all the participants, billing and security issues make P2P unsuitable, except at the moment in the case of internet matching games, such as Microsoft's Age of Empires.

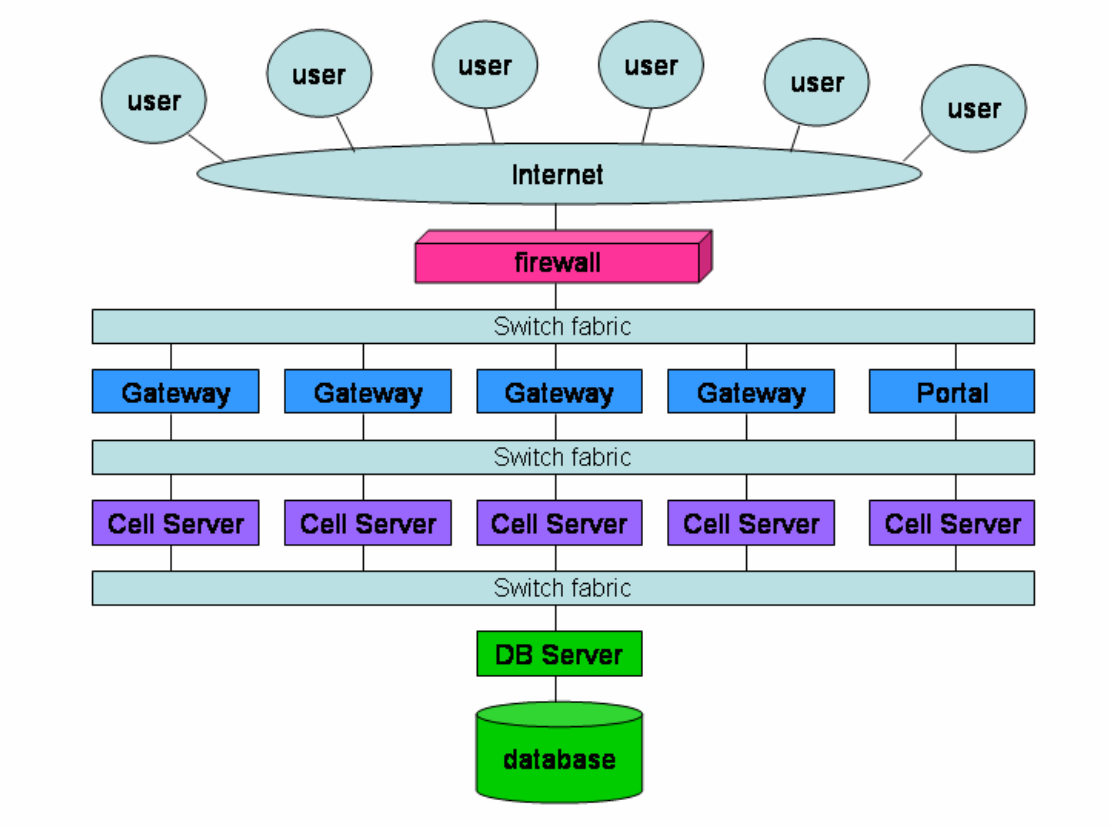

Figure 3-2. Generic 4-tiers architecture of MMOG

Other academics or open source project middleware for MMOG have also proposed similar architecture. Mauve [32] indicates that part of the server functions can be assigned to the proxy, which means that it can help the server, for example to distribute loading or prevent cheating. The proxy can also use specific hardware to improve scalability [11]. It can also be seen as an extension to the server, providing a flexibility of design and deployment, and the information coming from the proxy can be trusted.

The following is a summary of some of the technologies:

[Butterfly.net – Butterfly Grid]

While traditional MMOGs rely on a rigid centralized-server approach, Butterfly.net [7] aims at a self-managed, fully meshed network that shifts processing to idle resources

as needed. Butterfly.net made a strategic decision to build Butterfly Grid, which is based on the grid computing model. The grid consists of two clusters of approximately 50 IBM eServer xSeries servers. IBM technologies and services were selected by Butterfly.net grid, including DB2 Universal Database as the database software, WebSphere Application Server as the application server, and IBM Global Services for implementation and hosting. At a high level, these components are integrated into the Grid's fabric by Globus Toolkit.

[Microforté – BigWorld Technology]

Microforté spent over eight million dollars and three years of research and development on building the complete solution in the shape of "BigWorld Technology" [5]. That creation includes a scalable, reliable, customizable and fault-tolerance server infrastructure that can handle millions of players, a client 3D engine, and a set of tools to build and manage the MMOG world. Features of the BigWorld server include, adaptive load management, on the fly reconfiguration, and optimized RPC mechanism. According to web information, BigWorld seems to be the only vendor to deliver a total solution that covers everything from client 3D to server backend development.

[ICE]

ZeroC focuses on building a highly efficient middleware platform, called Internet Communications Engine (ICE) [35]that is as powerful as CORBA, without making all of CORBA's mistakes. In addition to avoiding CORBA's complicated core, it supports more of the features that aid MMOG development and maintenance, such as versioning of the object state, distributing software updates, efficiency protocol, defining the persistent support setting in a Slice IDL, and multi-language support. Also, Ice is used by a MMOG called Wish that supports tens of thousands of

simultaneous players.

[Massiv]

Massiv [33] is a distributed game middleware whose purpose is to simplify the development of distributed persistent MMOGs. The Programming Paradigms of Massiv are the same as with ICE, which is IDL-based. The most interesting part of Massiv is that it is designed to run on many servers located throughout the world. Therefore, issues of security, latency and time synchronization are addressed in Massiv.