DaCAP- A Distributed Anti-Cheating Peer to Peer Architecture for Massive Multiplayer On-line Role Playing Game

Huey-Ing Liu and Yun-Ting Lo
Fu Jen Catholic University, Department of Electronic Engineering
hueyingliu@gmail.com and 494506189@webmail.fju.edu.tw

Abstract

MMOG is one of the most popular amusements for people today and MMORPG is the hottest of all types of MMOGs. Due to the massive number of players, huge server load and player cheating are the two major problems of MMOGs. This paper studies the load and cheating issues of MMORPGs. Hybrid architecture by combining P2P architecture and client/server architecture is proposed for MMORPGs to reduce server load and improve scalability and flexibility. A fully distributed anti-cheating mechanism with players reciprocally monitoring is also designed in this paper. Analytical and simulation results demonstrate that the proposed architecture significantly reduces server load, increase system scalability, and efficiently detects cheating.

1. Introduction

With the development of network and communication technology, modern human life strongly links to Internet. Diverse activities including: entertainment, communications, shopping, educations, etc., are proceeded over Internet. Recently, a new style of online entertainment, named Massive Multiplayer Online Game (MMOG), which refers to those interactive games played by a lot of players in virtual space over Internet, gains great attentions. Statistical data shows that over 70% computer users have played at least one online game. Currently, average numbers of members of the major five MMOG in Taiwan are all over five millions. Basically, number of simultaneous online players reaches 50,000 in regular hours and hits over 300,000 during peak hours. However, there are two major problems for online games: server load and user cheating. To provide satisfied service quality, a considerable number of servers is required and significant installation costs are also produced. Cheating is the second major problem for online games, some tricky players may use illegal programs such as external programs to execute the game improperly or falsity the information then sale the virtual currencies and virtual items that obtained illegally. Such behavior seriously harms the agents and even ceases many games.

This paper investigates the major two problems: load and cheating for Massive Multiplayer Online Role Playing Game (MMORPG). To ease server load, peer-to-peer (P2P) architecture, named Dynamic anti-CheAting Peer to peer (DaCAP) architecture, is proposed for MMORPGs. In DaCAP, nearing players over virtual space self-organize into clusters and dynamically adjust cluster size according to the population of players. Cluster based processing significantly reduces the server load. In addition, an anti-cheating mechanism is also designed in DaCAP to prevent improper execution and falsity. Simulation results demonstrate that the proposed DaCAP efficiently reduces the server load and prevent cheating.

2. Related works

2.1. Internet games and online games

Multiplayer interactive games are classified into two categories: Internet games and online games. Through networks or the Internet, an Internet game player may either start a new game on his own computer or join the other's game via linking to a selected player's computer. The information of active games with their starters is obtained from a server. The server only records all active games with their starters but does not watch the whole process of each game. Different from Internet games, online games require specific servers to watch the whole process and record all players' actions. Thus, the server's jobs of online games are much more complicated than that of Internet games. There are many different types of online games such as sport, gambling, fighting and so on. Among all types of online games, role playing game (RPG) is the most popular one. Some popular online games attract over millions of users. With the requirements of serving millions of players, Massive Multiplayer Online Game (MMOG) is therefore developed. Still Massive Multiplayer Online Role Playing Game (MMORPG) is the hottest of all types of MMOG.

2.2. Architecture of MMORPGs

Most of current MMORPGs adopt client/server (C/S) model [1][2][3]. To serve massive players, multi-server is an often adopted architecture. The multi-server architecture [2] separates the servers with different functions such as patch, login, game and area servers as shown in Figure 1. Players first confirm game version with the patch server. Update is executed if the version is inconsistent because players must use the same latest game version. Then players must connect to the login server to verify their IDs and passwords. All regular operations of the game are controlled by game servers which undertake most of the loads. Therefore, game servers

978-0-7695-3156-4/08 $25.00 © 2008 IEEE
DOI 10.1109/CCGRID.2008.49

usually comprise several servers and balance loads among them to provide fast responses.

To further reduce load, two layered structure might be applied by further separating the game server into several area or channel servers. The whole game world comprises several small areas and players in each or few neighboring areas are controlled by an area server. While, channel servers divide game into several channels, players are free to choose a channel to enter and channel servers serve all the users in its channel. The famous game Maple Story is one of the channel division games, and the game Tales Weaver adopts area division method.

The advantage of this C/S model is centralized control which provides higher level of security for players' information and the game itself [4]. In addition, the structure is relative simple and both design of game and system allocation are easier [1]. However, Abdennour el Rhalibi and Madjid Merabtii claim that such architecture also suffers high installation cost, heavy maintaining effort, large bandwidth requirements and low flexibility [5]. In order to meet the demands of over 10 thousand players, several excellent servers must work together, which will increase cost of the servers and result in considerable maintaining effort. In addition, C/S architecture lacks excellent flexibility. For example the max number of online players is 50,000, the system must be equipped with capacity with 60,000 players. However, since only 25,000 players are online during non-peak hours, some resources are wasted.

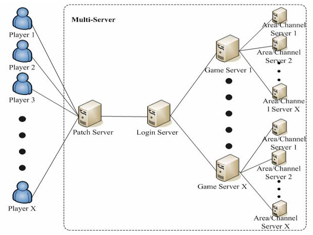

Figure 1. Architecture of multi-server system

In recent years, P2P emerges and becomes one of the most popular approaches for content distribution and data sharing over the Internet. In contrast with traditional C/S systems, P2P systems utilize upload capacities of all clients and therefore are superior in scalability, flexibility, and server-load reduction. The characteristics of P2P systems deliver huge attractions to shift online games from C/S model to P2P model especially for MMOGs [3][5][6]. P2P based MMOGs distribute server's load to players, thus significantly reduces server's burden and improves the scalability and flexibility. However, the capacity of P2P system purely relies on clients themselves, thus most P2P systems suffers instability, unfairness and cheating. Different from content sharing systems, instability or unfairness does not impose too much inference on P2P based MMOGs. However, cheating is the major problem of P2P based MMOGs, because most computations are done by players themselves, which make it much easier to cheat than traditional C/S architecture. In reference [5], a P2P architecture for MMORPGs that selects some players as supernodes is proposed. Supernodes take responsibility for managing parts of the players. A primary problem of supernode model is that if the supernode is an illegal player, significant impacts on the game's operation will be caused. Even though the supernode is a legal player, it is unfair for them to contribute computer resources without reward. Reference [7] refines the supernode model by selecting some players as monitors to watch supernodes. Such architecture provides better security; however, if both supernodes and monitors are cheaters, the system still falls into disaster. The anti-cheating mechanism proposed of DaCAP is in fully decentralized fashion, unfairness and minority control are efficiently prevented.

3. The system description of DaCAP

DaCAP, combining P2P architecture and C/S model, divides the playing region into several areas for efficient management. The Hotspot and Raid in DaCAP adopt the same management scheme of C/S architecture. The hotspot generally refers to areas with high player density, such as towns in a game because the player density and floating rate in town is usually higher than other areas. In addition, since behaviors of players such as trading or starting a quest in a town are usually sensitive and complicated, so players in a town are controlled by servers directly. Raid usually refers to an independent space created for a team of players to complete quests. Since players in the same Raid to conquer a quest are usually friends and there is higher possibility to cheat the system together. To prevent such a circumstance, Raids are also controlled by servers.

In DaCAP, the management among the other areas, say non-hotspot areas, adopts P2P architecture. Players in non-hotspot areas are organized into groups. Each group forms P2P overlay architecture, i.e., fully connected connections are built among all group members. Each time when a player joins, it distributes its initial status to all the other group members. The others also send their own current status to the new coming player. Instead of server control, players in a group are responsible for computing and monitoring the behaviors of all the players in the same group. Namely, the P2P group periodically synchronizes by its group members. Such reciprocally monitoring, illegal players are more difficult to cheat the system and therefore achieve defraud-proof. Furthermore, through players' self-monitoring and self-computing, load of the server is significantly reduced. DaCAP hybrid architecture is shown in Figure 2.

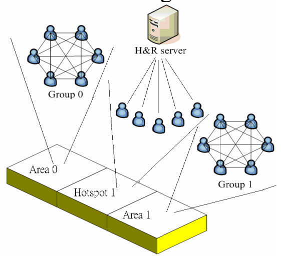

Figure 2. The hybrid architecture of DaCAP

3.1. System architecture

In DaCAP architecture, according to servers' functionalities, they can be classified into a Login server, Hotspot&Raid (H&R) server, Area server, Check server and database as shown in Figure 3.

Login server is responsible for confirming game version, and verifying ID and password of a player.

The H&R server, denoted as H&R server, handles the game computations of the players in hotspots and Raids. The function of the H&R server in DaCAP is similar to the gamer server of traditional C/S architecture.

The Area server manages the members of each group, controls area changing and acknowledges the associated member list to new coming players. Each non-hotspot new-coming player is directed to the Area server and receives a member list of its group (area). Meanwhile, the Area server also notifies all the other group members with the IP address of the new player. To manage all areas and their members, an area table, including information of ID, IP address of each non-spot player and his associated area ID, is maintained in the area server as shown in Table 1. Additionally, the Area server is also responsible for load balancing among areas which will discuss latter ins section 3.3. While a player leaving the group, all players in the group update their own records of changing data, including: currency, equipment and so on of all players in the group with the Check server.

The Check server verifies all the received data to check any inconsistency. Only consistent data is saved in the system's database. In addition, Check server is also responsible for recording the information of cheaters and informers, such as cheating players' ID, cheating count, and informer ID, informing count, etc., respectively. Database mainly keeps all players information such as IDs, passwords, IP addresses, area IDs, currencies, equipments and so on.

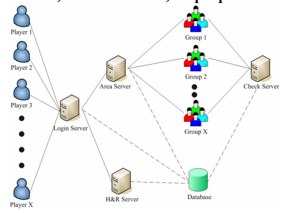

Figure 3. The system architecture of DaCAP

Table 1. The area table in Area server

Player ID	IP Address	Area ID
Player 1	10.10.16.6	1
Player 2	36.25.176.7	2
Player 3	140.36.147.7	3
Player 4	202.17.198.8	4

3.2. System Operations

After a player, say player A, completes the login procedure through Login server, previous logout position is used to determine whether player A is handled by H&R server or P2P model. If previous logout position was in a hotspot or a Raid, player A builds up a connection with the H&R server to maintain proper game operations. While the previous logout position is in a non-hotspot area, Login server would first direct the control of player A to Area server, and then Area server decides an appropriate area (group) for it based on its previous location. Afterward, player A will obtain a list of IPs of all the players in its group via Area server and the IP of the new player will also be notified to all the players in its group by Area server. With the group list, player A builds connections with all the other players in his group and notifies them with his initial status. Player A is therefore finished the joining process and starts the game under P2P mode.

During playing, if behaviors of player A are computed by H&R server, player A updates his status with the H&R server at a system defined synchronous-frequency, denoted as *f*. H&R server acknowledges player A of his surrounding environment simultaneously. All actions of player A are computed and supervised by H&R server, and the status of player A is saved in the system's Database. Otherwise if player A is in a non-hotspot area, he will play game under P2P model. According to Area server, player A is assigned to an appropriate group. Under P2P model, each player has to compute all the actions of other players in the same group and record the resulting statuses so as to determine the legality of their actions. Thus a local database is maintained in each non-hotspot area players to record the current statuses for his group members. Each action of a player is sent to all the other players for computation and verification. In addition, group members also periodically synchronize with each other at the synchronous-rate *f*. In case of finding any illegal action of a player, the cheating player is reported to the Check server. Whenever a player leaves, all players in the same group update all statuses in their databases with the Check server.

While a player A is about to cross the area boundary, a leaving message is either broadcast through P2P architecture or sent to the H&R server according to the adopted control model so to notify his leaving. If player A is in a non-hotspot area, then all players in his group synchronize their database with the Check server and update the group list accordingly. Once receiving the update, the Check server verifies the legality of all group members and then updates the system Database. In the meantime, player A also sends an area-changing message to the Area server to change area. Upon receiving the area-changing message, the Area server determines whether he roams to a non-hotspot or not. If player A is in a non-hotspot area, the Area server assigns a proper area (group) for player A and notifies him with the new group list. Meanwhile, the information of player A is also sent to his new group members for group list update. If the new area of player A is a hotspot, the Area server only has to notify both player A to switch to C/S model and then player A builds a connection to the H&R server to proceed his playing. The area-changing process of player A is then finished.

However, if the original location of player A is a hotspot area, player A sends a leaving message to the H&R server and Area server. The connection between player A and the H&R is then closed. If player A roams to a hotspot area, the Area server notifies player A with a new hotspot area ID. Player A then reconnects to the H&R server with his new hotspot ID so as to proceed proper game playing. However, if player A roams to a non-hotspot area, the Area server has to determine a proper area for player A and then notifies player A to switch to a p2p mode and his group list. Moreover, the information A is also sent to the group members of player A.

3.3. System load balancing

In DaCAP, all members of a P2P group have to compute and record all actions and statuses of their members. In DaCAP, group size affects the system load, anti-cheating ratio, and it also impacts players' load. Thus, while the number of players in a non-hotspot area exceeding a predetermined upper bound B_{up}, the Area server divides the crowded area into two sub-areas. While the number of players in a non-hotspot area is lower than the system defined lower bound B_{low}, the Area server combines the area with its adjacent areas to form a super-area. The dividing and combination of areas are shown in figure 4. In order to avoid wasting too much resources in division and combination, the area changing time interval T is defined and the interval between division and/or combination must be larger than T.

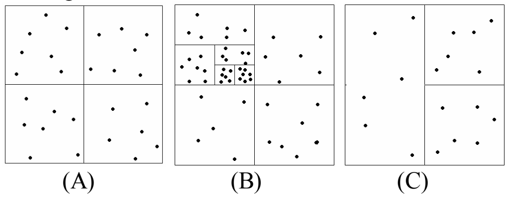

Figure 4 An illustrated example of dividing and combination of areas; (a) equal-sized sub-areas under evenly distributed players, (b) high density sub-area split into several sub-areas, and (c) sparse areas merged into a super-area.

3.4. Anti-cheating mechanism

Distributed and centralized anti-cheating schemes are adopted in DaCAP according to the used computation modes. When players are controlled by H&R server, centralized behavior computation and anti-cheating is executed by the H&R server. Under P2P mode, distributed anti-cheating, which players themselves watch mutually, is adopted. If a cheating player A uses illegal program to play game under C/S mode, the H&R server judges that player A for illegal behavior and kicks off player A from the game. While player A uses illegal program to play game in P2P mode, the cheating behavior of player A is detected by his legal group members. Once cheating behavior is detected, the cheating player is reported to the Check server. Meanwhile, player A is kicked out.

In addition, whenever a player leaves the group, all statuses of all players in the group are updated to the Check server. The Check server then verifies all the collected data. All consistent data is updated to the system database. However, any inconsistent data appears, it is handled as follows.

1. If the inconsistent data regarding to player A is reported by himself, player A will be marked as a falsify player.
2. If the inconsistent data regarding to player A is reported by player B, player B will be marked as informer. In addition, the status of player A stays in previous state, i.e., all the actions of player A are invalid since last update.

Table 2 shows the records send from different players of same group to the Check server. In the record from player A, the amount of player A's gold is 1,000,000, but in the record from player B and C the amount of player A's gold is 100. In this case, player A is marked as a falsify player and player B and C are marked as informers. The data of player A is not going to be updated to the Database. And for later analyze system will keep logs of cheating and informing shown in table 3.

Table 2 Records from different players of same group

Record from player A						
Player ID	HP	MP	EXP	Gold	Time	Date
Player A	69	97	415	100,000	16:03	2007/12/14
Player B	87	28	648	500	16:03	2007/12/14
Player C	93	62	232	600	16:03	2007/12/14

Record from player B						
Player ID	HP	MP	EXP	Gold	Time	Date
Player A	69	97	415	100	16:03	2007/12/14
Player B	87	28	648	500	16:03	2007/12/14
Player C	93	62	232	600	16:03	2007/12/14

Record from player C						
Player ID	HP	MP	EXP	Gold	Time	Date
Player A	69	97	415	100	16:03	2007/12/14
Player B	87	28	648	500	16:03	2007/12/14
Player C	93	62	232	600	16:03	2007/12/14

Table 3 Logs of cheating and informing

Cheater ID	Informer ID	Time	Date
Player A	Player B	16:03	2007/12/14
Player A	Player C	16:03	2007/12/14

Cheater ID	Cheat count
Player A	2

Informer ID	Inform count
Player B	1
Player C	1

Besides, to avoid collaborated falsification of player data, the system randomly chooses a number, denoted as n_r, of players from other areas to join the group so as to prevent illegal players from collaborated cheating. As long as there is one legal player in the group, cheating is avoided and falsification is able to be detected. In MMORPGs, it is very difficult to find a group without any legal player.

In addition, to avoid malicious informing against other players, all informers and their informing times are recorded in the system. Once the informing counter reach a predetermined threshold, the associated informer is regarded as a malicious informer and is punished according to the system defined principles such as lock the account of malicious informers and cheaters. Such system can detect and prevent cheating, (collaborated) falsification, and malicious informing efficiently.

4. Effectiveness Evaluations

In this section, to understand the performance of the proposed DaCAP, computer simulations are implemented. Simulation results of DaCAP are compared with that of traditional C/S architecture.

4.1. System performance analysis

Assume the number of players in the system is N. All players are classified into two categories: hotspot players with a number of N_h and non-hotspot players with a number of N_{nh}, where $N=N_h+N_{nh}$. The concentration ratio of players, denoted as α, defined as the ratio of the number of hotspot players to the number of all players, i.e., $\alpha=N_h/N$. A floating rate of players, denoted as β, is defined as the ratio of changing areas players per second. Thus, αN represents the number of hotspot players, and $2\alpha NP$ represents number of message transmission between players in hotspot and the H&R server. Non-hotspot area players can be represented by $(1-\alpha)N$. Assume the average group size is G. Since a message size of G is transmitted by all players in non-hotspot to the Check server while a player leaves the group, $\beta(1-\alpha)NG$ represents the number of message transmission between non-hotspot area players and Check server. Number of players that are shifting between areas is represented by $N\beta$. Since players list of size G is sent by Area server to the area-changing player, a total number of $N\beta G$ messages are sent by the Area server while area changing. Hence, the traffic load of system servers, denoted as L_t, is determined by the following equation.

$$L_t = 2\alpha Nf + (1-\alpha)\, N\beta G + N\beta G + N\beta$$

There are three major tasks that server fulfills during the game. The first task is controlling hotspots and Raids, the second is to check the consistency of collected players' status, and the third is to determine a proper group for non-hotspot players. Without loss of generality, assume the computation-load ratio of per computation of these three tasks is $1{:}w_1{:}w_2$. Notably, since the average group size is G, thus for the second task G times of consistency checking is performed. Hence the effectiveness load of system serves, denoted as L_c, is determined by the following equation.

$$L_c = \alpha Nf + w_1G(1-\alpha)N\beta + w_2\, N\beta$$

Table 2. Summarize the system parameters.

N	Total numbers of players
N_h	Number of players in hotspot
N_{nh}	Number of players in non-hotspot
α	Concentration ratio of players
f	Synchronization rate
G	Average group size
β	Floating rate of players
L_t	Traffic load of system servers
L_c	CPU load of system servers
B_{up}	Upper bound of number of players in an area
B_{low}	Lower bound of number of players in an area
T	Interval of consecutive division or combination
n_r	The number of randomly selected group players

4.2. Simulation results

The simulated game space is 300 × 600. With different concentration ratios, players are located in either in hotspots or non-hotspot areas with different probabilities. The virtual game space is further divided into 50 areas of size 60 × 60. Five randomly selected areas are hotspots. The roaming speed is assumed to be one pixel per second. There are 8 possible directions. Random roaming model is implemented, thus a moving player randomly chooses one of eight directions to move. Initially, players with 50% probability move, i.e., players with 50% probability stand still. In the period of moving, with 80% probability the players continue moving and 20% probability they stop moving. For standing-still players, with 70% probability the players keep stay at the original place and with 30% probability they move. Players return back in case of reaching the edge of the game space. The selected simulation parameters are summarized in the following table.

Table 3. Simulation Assumptions

Simulated space	300*600 pixel
Player size	1 pixel
Player roaming speed	1 pixel per second
f	4 times per second
B_{up}	55 players
B_{low}	25 players
G	45 players
n_r	5 players
T	30 seconds
Massage size	1000 bits
Simulation time	300 seconds

Figure 6 shows the traffic loads of DaCAP and traditional C/S architecture under different concentration ratios. The solid curves show the results obtained by analysis, and the dotted lines represent the results of Simulations. As shown, the number of transmissions of DaCAP is significantly lower than that of c/s model. The comparison of the CPU loads obtained by traditional C/S architecture and DaCAP with different concentration of players is shown in Figure 7. The CPU load of DaCAP is also much lower than that of C/S model. In addition, the increasing slop of DaCAP is also smaller than that of the c/s model, thus DaCAP provides higher scalability.

Different group bonds are simulated so as to understand the influence of group size on the system performance and anti-cheating ratio. Figure 8 shows the anti-cheating ratios versus different group sizes. As shown, the larger group sizes the higher anti-cheating ratio. Even if over 90% of players cheat, DaCAP is still able to detect cheating with very high ratio. The traffic and CPU loads under different group sizes are shown in Figure 9 and Figure 10, respectively. Since only server load is considered, thus similar loads are obtained under different group sizes. However, a larger group produces higher loads under P2P architecture.

To understand the inferences of roaming speed, in Figure 11 and Figure. 12, the simulation results of traffic and CPU loads are shown under different roaming speeds. Obviously, higher roaming speeds produce higher traffic and CPU loads.

To understand the system scalability of DaCAP, the simulation results of bounds of maximum players under different server computation power and bandwidth are shown in Figure 13 and Figure 14. As shown, DaCAP provides significantly higher scalability in terms of computation and bandwidth capabilities.

5. Conclusion

This paper proposed hybrid architecture for MMORPGs. Since the power of PC is much stronger and the bandwidth of networks is higher than ever, the proposed DaCAP shifts load from server to players so as to enhance the scalability and flexibility of game systems. The framework of DaCAP not only reduces the system server loads but also provides a good nature for anti-cheating. The simulation and analytical results demonstrate that the DaCAP significantly improve the system performance in terms of CPU and traffic loads. Furthermore, high anti-cheating probability is also supported in the proposed DaCAP.

6. References

[1] Sergio Caltagirone, and Matthew Keys, "Architecture for a Massively Multiplayer Online Role Playing Game Engine", Journal of Computing Sciences in Colleges Volume 18, Consortium for Computing Sciences in Colleges, USA, December 2002, pp. 105-116.

[2] Tobias Fritsch, Hartmut Ritter, and Jochen Schiller, "The Effect of Latency and Network Limitations on MMORPGs (A Field Study of Everquest2)", the 4th ACM SIGCOMM workshop on Network and system support for games NetGames '05, ACM, New York USA, October 10-11 2005, pp. 1-9.
[3] Shin Ito, Hajime Sogawa, Hiroki Saito, and Yoshito Tobe, "A Propagation of Virtual Space Information Using a Peer-to-Peer Architecture for Massive Multiplayer Online Games", the IEEE ICDCSW'06, IEEE, July 2006, pp. 44-44.
[4] Patric Kabus, Wesley W. Terpstra, Mariano Cilia, and Alejandro P. Buchmann, "Addressing Cheating in Distributed MMOGs", the 4th ACM SIGCOMM workshop on Network and system support for games, ACM, New York USA, October 10-11 2005, pp.1-6.
[5] Abdennour El Rhalibi, Madjid Merabti, "Agents-Based Modeling for a Peer-to-Peer MMOG Architecture", Computers in Entertainment Vol. 3, ACM, New York USA, April 2005, pp. 3-3.
[6] Thorsten Hampel, Thomas Bopp, and Robert Hinn, "A Peer-to-Peer Architecture for Massive Multiplayer Online Games", the 5th ACM SIGCOMM workshop on Network and system support for games, ACM, Singapore, October 2006.
[7] Takato Izaiku, Shinya Yamamoto, Yoshihiro Murata, Naoki Shibata, Keiichi Yasumoto, and Minoru Ito, "Cheat Detection for MMORPG on P2P Environments", the 5th ACM SIGCOMM workshop on Network and system support for games, ACM, Singapore, October 2006.

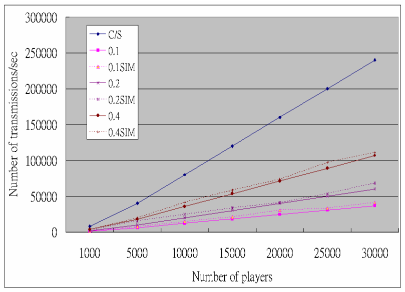

Figure 6. The analytic and simulation results of traffic load with different concentration ratios.

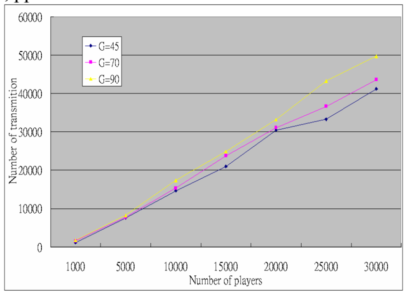

Figure 9. The traffic loads versus different group sizes

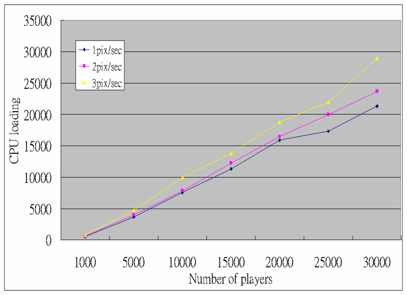

Figure 12. The CPU load with different roaming speeds

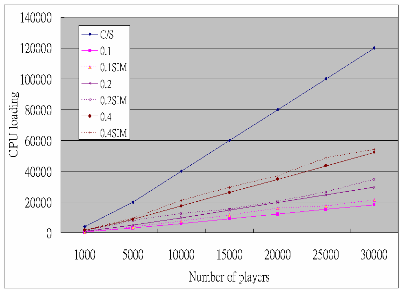

Figure 7. The analytic and simulation results of CPU load with different concentration ratios

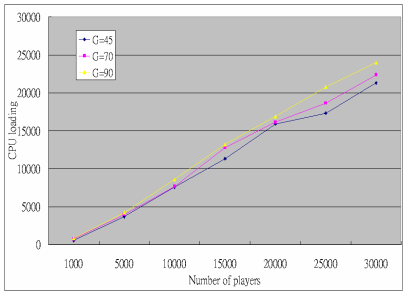

Figure 10. The CPU loads versus different group sizes

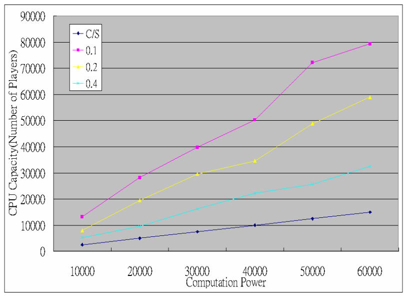

Figure 13. The number of maximum affordable players of DaCAP versus computation power

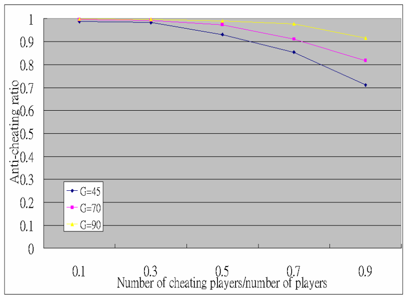

Figure 8. Anti-cheating ratio with different group sizes

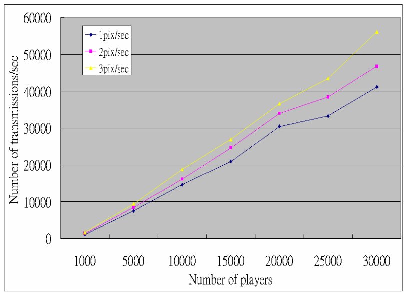

Figure 11. The traffic load with different roaming speeds

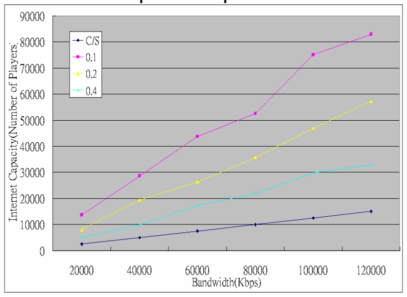

Figure 14. The number of maximum affordable players of DaCAP versus bandwidth