PAPER *Special Issue on New Technologies and their Applications of the Internet*

HYMS: A Hybrid MMOG Server Architecture*

Kyoung-chul KIM[†a)], **Ikjun YEOM**[†b)], ***and*** **Joonwon LEE**[†c)], ***Nonmembers***

SUMMARY The massively multiplayer online game (MMOG) industry is suffering from huge outgoing traffic from centralized servers. To accommodate this traffic, game companies claim large bandwidth to Internet Data Centers (IDCs), and several months' payment for that bandwidth is likely to even exceed the cost for MMOG servers. In this paper, we propose a MMOG server architecture to reduce outgoing bandwidth consumption from MMOG servers. The proposed architecture distributes some functions of servers to selected clients, and those clients are in charge of event notification to other clients in order to reduce the outgoing traffic from servers. The clients with server functions communicate with each other in peer-to-peer manner. We analyze traffic reduction as a function of *cell-daemonable* ratio of clients, and the results show that up to 80% of outgoing traffic from servers can be reduced using the proposed architecture when 10% of clients are cell-daemonable.

key words: *MMOG, server, peer-to-peer, hybrid, traffic*

1. Introduction

Online games have in many ways revolutionized the interactive entertainment industry and have created an important new source of revenue and gamers. Most industry analysts agree that online games will continue to grow in raw usage, and will generate significant revenue, with predictions varying from $1.8 billion in 2005 to $2.55 billion in revenue in 2006 [1]. According to [1], massively multiplayer online games (MMOG) market is regarded as one of most profitable segment of whole online game market.

There are two different schemes to implement online games: client-server and peer-to-peer. Most of MMOGs adopt client-server scheme that provides advantage on consistency, persistence and administrative control [2]. The peer-to-peer scheme generates problems about consistency, persistence and security because each client has its own copy of the game status which is updated by the messages coming from other clients, and there is no central repository for persistent data.

Manuscript received March 31, 2004.
Manuscript revised June 25, 2004.
Final manuscript received July 30, 2004.

[†]The authors are with the Department of Electrical Engineering & Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, Korea.
a) E-mail: ddaeng7@camars.kaist.ac.kr
b) E-mail: yeom@cs.kaist.ac.kr
c) E-mail: joon@cs.kaist.ac.kr

[*]This work was supported by ITRC program funded by Ministry of Information and Communication, Korea

Without any treatment, total amount of traffic is identical in both schemes. When a player generates a message, the message should be delivered to all the other players either through a centralized server or by peer-to-peer network. IP-multicast is an attractive alternative but it has been far from wide deployment due to its intrinsic drawbacks [3].

Client-server scheme needs scalable server and huge bandwidth because the traffic is centralized on the server only, even though it is possible to reduce traffic by several methods such as packet aggregation, interest management [4] and so on. There are several approaches to distribute traffic among multiple servers such as mirrored server [2], [5], generic proxy [6], [7] or booster box [8]. These approaches reduce the latency only. The total sum of traffic is equivalent at least, and so is the summation of distributed bandwidth-charge in Internet Data Centers (IDCs).

Computing speed and memory capacity has increased following the Moore's law. The bandwidth of client also has increased with increasing broadband internet services based on cable modem and xDSL technologies. South Korea has the largest broadband deployment per capita in the world, where about 60% of households are wired [1]. There is increasing extra resources on client machine such as CPU idle time, free memory and network bandwidth because most of MMOG was designed to operate on mid-ranged client machines via 33.6K modem.

In our experience about MMOG, it was possible to accept about 3,000–4,000 concurrent players on a machine that equips dual 2.4GHz Xeon processors and 2Gbytes of RAM, and its price is about $1,500 at Mar. 2004. The whole server cluster composed of 20 dual processors machines can probably accept about 60,000–80,000 concurrent players. Assuming maximum traffic in normal operation is 50% of 100Mbps Lan, it may be needed 1Gbps for 20 machines in normal operation.

It may cost $40,000–$50,000 for a MMOG server cluster composed of 20 machines at the beginning, but the network cost per month is around $23,000 [9]. Even though people argues only about scalability and seamlessness of server cluster, in reality, the network cost is the most important issue that has to be dealt with.

In this paper, we address the issues related with server-side network traffic, and propose a hybrid architecture such that: the server system is organized with

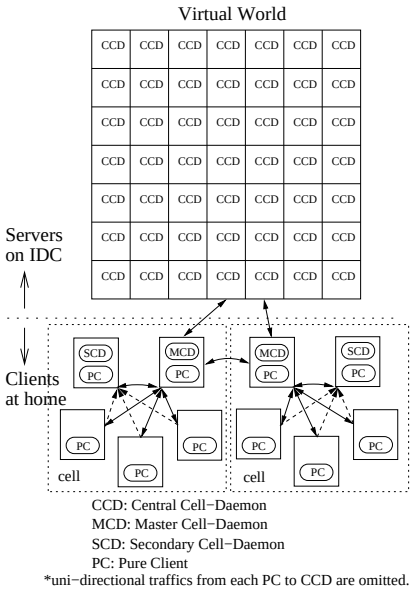


Fig. 1 HYMS architecture

the central server and multiple clients which are selected among qualified clients and are connected each other with peer-to-peer manner. While the central server maintains the whole game, the selected clients are in charge of communication with game players to save the central server's outgoing bandwidth. Meanwhile, between game players and the server system, the client-server relation is maintained to handle consistency, failure and security problems.

2. Approach

Even though peer-to-peer architecture reduces significantly the outgoing traffic from the central server, it is vulnerable to crashes which lead to inconsistent game status and service outages. Therefore, we suggest a hybrid architecture where a client machine becomes a local game server when it is equipped with enough CPU power, memory and network bandwidth. To cope with crashes, the central server has a replica of the game status maintained by each local game server.

Fig. 1 illustrates the HYMS architecture. The virtual game world is divided into multiple cells and each cell is administrated by a daemon process running at a central cell server. A CCD in the figure is such a daemon process. The size of cell is chosen as to accommodate a proper number of players, and the cell can be divided further recursively, if needed. When a player crosses cell boundary, a handoff from one CCD to another CCD occurs. So far, it is the same as the traditional client-server architecture.

Our scheme differs in clients since a client machine takes up the role of a CCD whenever possible. At the bottom of the figure, client machines are shown with different roles. An MCD is a daemon process running

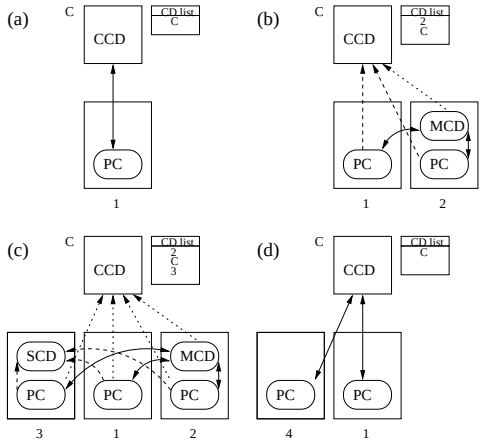


Fig. 2 snapshots of HYMS in operation

at a client machine, whose role is similar to the one of CCD, and the first qualified client machine among the clients connecting to a CCD hosts the MCD process. An SCD is a back-up process for an MCD. Usually, a machine with enough CPU and network capability that has connected after MCD hosts SCD, hence there may be several SCDs per cell. An SCD becomes the MCD of the cell if the MCD crashes or logs off. To be a back-up process of an MCD, the SCD should be informed of exactly the same events as the MCD. It is possible that no one in the cell is capable of the MCD, and the CCD should be the cell master as in the traditional client-server architecture. For this purpose, each CCD is also updated by the corresponding MCD periodically, and CCD does not act to client's actions, hence acting like an SCD. A PC is a pure client that is the same as a client in the client-server architecture but connects to the MCD rather than to the CCD. Each PC notifies its actions to the corresponding SCD and CCD as well as the MCD to maintain consistency.

Fig. 2 shows operations between HYMS components. A CCD maintains a list of client machines to keep track of which one is an MCD or an SCD, and the number of PC's in the cell. Fig. 2(a) denotes the case when a CCD administrates the cell since none of connected PC's is capable of becoming an MCD. When a new node with enough capability joins the cell, the node asks the CCD for the grant to be an MCD of the cell. The CCD decides whether this new node can be an MCD or not based on its capability and the number of clients playing in the cell. Fig. 2(b) shows the case when a new node becomes an MCD of the cell and administers the cell. Once an MCD exists for a cell, other nodes joining the cell with enough capabilities become SCD's as shown in Fig. 2(c).

3. Related Issues

There are many issues in adopting peer-to-peer scheme to MMOG, and most of them are solved by the na-

ture of HYMS. No consistency issues are raised because there is always only a single master CD per cell. There is no serious persistence issues either since each CCD is updated by the MCD. If a CCD cannot guarantee enough persistence of game status, it is an issue that is not peculiar to our scheme but to all the other game servers. In this section, we discuss several other issues raised from our peer-to-peer aspect of HYMS, and their solutions.

3.1 Client crash

The server program is running in a well-tuned environment like well-known hardware specification, proper OS, libraries and device drivers. But, a configuration of client machine covers a wide spectrum of hardware, and it is very hard to make client program reliable on every possible combinations of CPU, RAM, sound devices, graphic devices, OS, libraries, device drivers, and so on. Even though game machines like Xbox and Play Station are much more stable than personal computers, they still suffer from external dangers such as sudden power/network outage or physical harm that are not very rare.

Since crash of a pure client is none of our concern, we confine ourselves to the case when MCD crashes. A CCD, MCD and SCD receive packets from PCs, and only the MCD sends packets to PCs. For efficient and reliable backup, checkpointing and incremental logging is applied to SCD and CCD. An SCD and a CCD collect any change in each client for logging and collect checkpoint snapshot from the MCD periodically. The amount of traffic for checkpointing generated from each MCD is proportional to the number of PCs in the cell, and the number of MCD is limited by the number of cells. Traffic to the central server for checkpointing can be controlled by adjusting the checkpoint interval.

When MCD crashes, CCD can detect the crash by timeout since CCD is updated periodically by the MCD. Then, the CCD assigns the role of master to a SCD that is the second CD on the list, and the SCD becomes a new MCD. The new MCD recovers cell status using last checkpoint information that has been received from old MCD, and logged packets received from PCs after the checkpoint. New MCD announces its advent to all the clients in the cell. If no node in the cell is capable of being MCD, the CCD becomes the cell master as shown Fig. 2(d).

3.2 Load balancing

In client side, upward bandwidth of ADSL is 256–640kbps, and this bandwidth limitation is imposed by the policy of internet service providers (ISPs) even though there exist technologies that can mitigate such limitation. In our experiences, a conventional client-server structured MMOG server needs about 10–15kbps downward bandwidth per client in average. Therefore, a single MCD can accommodate 15–60 clients.

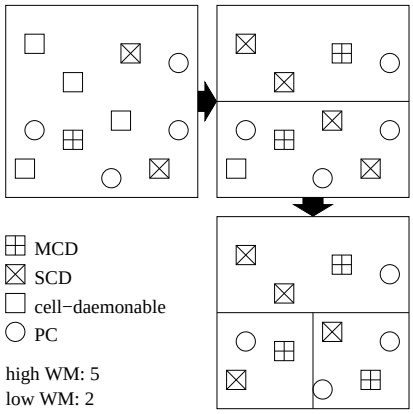


Fig. 3 recursive cell division

A cell should be divided into several sub-cells if it is too crowded. Cell division scheme depends on the cell shape, but most issues raised by our proposed P2P architecture are immune to cell shape. Therefore, we assume a grid shape as shown in Fig. 1 without loss of any generality. A cell is divided into two sub-cells when there are too many game players in the cell, with recursive division alternating vertical and horizontal, like 2-dimensional kd-tree [10], but divide virtual dimension evenly. In each step, a cell divided into only two sub-cells for fast divide and merge of CDs list(s) and CCD(s).

There are low watermark for merging and high watermark for division, and the low watermark must be smaller than half of high watermark to prevent repeated dividing-and-merging ping-pong. Fig. 3 shows the recursive cell division when the high watermark value is set to 5 and the low watermark value to 2. When a cell is divided into two sub-cells, new MCD may be selected among cell-daemonable clients that reported cell-daemonable to the CCD, but still doesn't receive grant packet. A CCD must maintain not only CD list but also the cell-daemonables list. In Fig. 3, it is shown that new MCD and SCD are selected from cell-daemonables at each division. The CCD is also divided, i.e, the old CCD forks a new CCD on a proper machine in server cluster. If there is no cell-daemonable client in a new sub-cell after division, the new CCD becomes MCD.

3.3 Latency and network traffic

The network latency between an MCD and CDs may increase since they may belong to different ISPs. The number of hops between them may increase, and so does the possibility of existence of slow router or network saturation. Large network latency can be an obstacle to player's interaction in multiplayer online game.

Many researchers report that some amount of net-

work latency is tolerable for multiplayer computer game since humans do not notice such latency. The observed value is about 100–500ms in [11], and the network latency of recent broadband internet services is almost one order smaller than this value [12]. However, there can be network sectors where load can be excessive. Traditional approaches to reduce network loads usually try to move network traffic to between clients. In *end system multicast* [3], the central server sends status update packets to a single client in a cell, and the client forwards the packet to others in the cell. A multicast (not IP multicast) hierarchy could be made in the cell to spread the load to several clients in the cell.

Our HYMS architecture reduces the network loads along the same line of the above approaches. Furthermore, while the above scheme experiences large latency in interactive applications such as MMOG because it has multi-layered hierarchy from the central server to terminal clients, our scheme reduces the latency since no more logical hops exist between an MCD and PCs. An MCD interacts with each PC in the cell, and thus it is possible that a client may experience larger number of hops to reach the MCD than to reach the central CCD. This observation leads us to investigate methods to divide cells considering the locations of clients in the cell. Since clients using the same ISP usually connected with small latency, this factor should be considered when a cell is divided and when a new MCD is selected.

In our scheme, only qualified client machines can be a candidate for MCD. If there is no candidate for MCD, a central CCD remains as the cell master. By injecting proper function into the qualification, it would be possible to estimate latencies that clients will experience from a new MCD. A new MCD selected though this process will guarantee average latencies.

3.4 Cell handoff

When a client crosses cell boundary, it should connect to a new server, and it is called *client handoff*, and most seamless MMOG servers treats this event without much hassles. With our scheme, since a client in a cell can be a server, a server machine may cross the cell boundaries, and this event is called *cell handoff*. SCD is a process prepared for this case. If no node in the cell qualifies for SCD, the central CCD becomes the cell master.

The leaving MCD become a normal client in a new cell. Since it was qualified for SCD and MCD in the old cell and this qualification may be valid in the new cell, it may become an SCD or MCD in the new cell.

3.5 NAT/Firewall

NAT(network address translation) is used to overcome the shortage of address space of IPv4. Though IPv6 promises much larger address space, it is still years away to be everywhere. A node in NAT environment cannot be an MCD because it is not given a fixed IP address.

To find out if a client is in NAT environment, we perform a simple test when each client joins the game: The client sends to the server a packet containing its local IP address. With simple comparison of the local IP address and the actual IP address in the IP header of the packet, the server can detect clients with global IP addresses.

Firewalls on ISP block specific ports in generally. For example, TCP port #80 is blocked commonly to prevent home user from operating web server on his/her machine. Some of other ports may have been blocked for any other reason of ISP. To workaround, wide range of candidate ports, and bidirectional port availability tests between server and client are need at joining. Only the nodes that pass this test become candidates for MCD and SCD.

Although the number of cell-daemonable clients is limited by two tests above at joining, an analysis in Section 4 shows that HYMS architecture is useful even with low ratio of cell-daemonables.

3.6 Cheating/Security

Pure peer-to-peer scheme is fragile from client cheating, packet tampering and so on. In HYMS, CCD can verify validity when receiving periodic checkpoint packets from the corresponding MCD. A CCD monitors its cell status using packets from PCs, and compares its own status with the checkpoint from the MCD to certify effectiveness of MCD. If an invalid status are detected from a MCD, the CCD regards the MCD as compromised and issues a new CD list without the current MCD to all PCs to inform the change of the MCD.

The IP addresses of MCDs and SCDs need to be exposed to other players in the same cell in HYMS, and it may be a security threat. At least, however, they know that their IPs are exposed, and thus may cope with the threat. Similar situation happens in the most peer-to-peer systems.

3.7 Facility of management

A CCD has full functions for a server as in traditional client-server scheme to cope with the case when there is no cell-daemonable client in the cell. There are additional management facilities required in HYMS: managing the current list of cell-daemons, grant/reject request for CD from clients, announcing new CD-list when it is updated, and managing a list of the cell-daemonables for recursive cell division.

4. Analysis

In this section, we analyze the impact of HYMS on traffic through simple models. Here note that we focus on outgoing traffic from the server rather than incoming

Table 1 variable definitions

l	length of square cell edge
r	radius of focus and nimbus
M	moving speed of a PC in virtual world
F	the ratio of moving PCs among all PCs.
D	density of PCs in virtual world, i.e. number of PCs in a circle with radius r
N	number of PCs in l by l cell
P	probability of existence of MCD in a cell
R	cell-daemonable ratio of clients

traffic. The amount of incoming traffic to the server in HYMS is mostly the same as in the traditional client-server architecture except traffic generated for checkpointing, which is controllable as described in Section 3.1.

The amount of outgoing traffic of central server depends heavily on the ratio of cell-daemonable clients among all clients in virtual world. In this section, we analyze the relation between the cell-daemonable ratio and the amount of outgoing traffic.

4.1 Modeling

The total outgoing traffic of each CCD is given by equation (1). It expresses combination of disjoint two cases whether a CCD is the master or not. P is the probability that there exists a cell-daemonable node in the cell, and T_{pp} is the traffic incurred in P2P mode, most of which is the traffic between CCD and MCD. Since most game traffic is between MCD and clients, this metric should be small. The probability that there is no candidate node for MCD is $1-P$, and CCD must act as a cell master like conventional client-server architecture. T_{cs} is the traffic for this case. Since a cell master should talk to neighbor cell masters and there is a chance that neighbor cell master is at a client machine. The extra outgoing traffic generated from this case is T_a. The traffic for handoff, T_h, must be also included in outgoing traffic when a master CCD is working on a cell. Note that traffic generated for client handoff from MCD to CCD is incoming traffic to CCD that is not an issue for a game server.

$$T_{all} = (T_{cs} + T_a + T_h)(1-P) + T_{pp}\ P \tag{1}$$

Each square cell with dimension of $l \times l$ divides the virtual space. Each PC in virtual space can have divided area of interest: *focus* and *nimbus* [13]. But, it is assumed that a PC's focus and nimbus is same circle with radius r. It is natural that when a man sees a woman, she can see him, too. This symmetrical visibility can be accomplished by setting both focus and nimbus of a PC to identical area. The density of PC D is defined by the number of PCs in a focus and/or nimbus. Variable definitions are listed in Table 1.

In this circumstance, when a PC sends a packet to inform change of its position, MCD sends about D packets to near PCs to inform the change of position. Even if the PC is near the border of the cell, for seamlessness, packets must be sent to and received from other PCs within distance r in virtual space, maybe in adjacent cells.

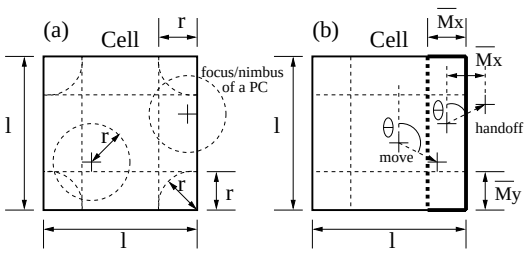


Fig. 4 (a) borders of a cell and area of interest of PCs, (b) border area for handoff

The outgoing traffic of CCD with no MCD is similar to case of conventional client-server architecture, and it is proportional to production of number of PCs in the cell area and the average number of PCs in one's nimbus, with constant C_{cs}.

$$T_{cs} = C_{cs} N D \tag{2}$$

Traffics from master CCD to adjacent MCDs comes in 8 directions: north, east, south, west and other 4 diagonals. Fig. 4(a) shows the borders of a cell and the areas of interest of two PCs. By the area of borders, PC density D, and P of adjacent cell, T_a can be expressed as

$$\begin{aligned} T_a &= C_a\ (4\frac{rl}{l^2}NP + 4\frac{1}{4}DP) \\ &= C_a\ (4\frac{r}{l}N + D)P \end{aligned} \tag{3}$$

For handoff, it needs to model how many PCs cross the boundary of a cell in unit time. Moving PCs may have virtually equal velocity with random directions. M means the 'moving distance per second' of a moving client in virtual world. M can be considered as a vector (M_x, M_y). The component M_x moves a PC to east-and-west, and the component M_y moves a PC to north-and-south. The average of component M_x to east side of randomized M can be written as following, when θ is 0 at the north and growing clockwise:

$$\overline{M_x} = M\frac{\int_0^{\pi} sin\theta d\theta}{\int_0^{2\pi} d\theta} = \frac{M}{\pi} \tag{4}$$

Not every PC is moving. F is the ratio of moving PCs. Since moving direction of PC is assumed random, the number of PCs crossing any side of a cell is same for all directions. The average number of PCs crossing east side boundary is the average number of moving PCs in $\overline{M_x} \times l$ rectangle at the east side in Fig. 4(b), because they will move for distance $\overline{M_x}$ to east in unit time. The overall average number of handoff on a cell in unit time becomes

$$\begin{aligned} \#\ of\ H.O. &= 4 \times (\#\ of\ handoffs\ to\ east\ side) \\ &= 4 \times (\#\ of\ moving\ PCs\ in\ (\overline{M_x} \times l)) \\ &= 4 \times F \times \frac{N}{l^2} \times \frac{M}{\pi} l = 4\frac{FNM}{\pi l} \end{aligned} \quad (5)$$

With density D, the handoff traffic, T_h, can be expressed with a constant C_h as follows:

$$T_h = 4C_h \frac{FNMD}{\pi l} \quad (6)$$

The outgoing traffic of CCD with MCD is only for meta-action: join, leave and crash recovery. Packets for meta-action are sent only to the client that request meta-action, not to near D clients in nimbus. Packets for normal operation, like movement, emotion, chatting, combat and so on, are generated by MCD on a client and it does not consume outgoing bandwidth of CCD. The amount of meta-action traffic is directly proportional to number of PCs in the cell area, with constant C_{pp}.

$$T_{pp} = C_{pp}\ N \quad (7)$$

R is ratio of cell-daemonable client. If there are N PCs in a cell with cell-daemonable ratio of R, the probability that there exists an MCD in the cell is

$$P = 1 - (1 - R)^N \quad (8)$$

D is the number of PCs in a circle with radius r, and N is the number of PCs in a square with length l sides.

$$N = \frac{l^2}{\pi r^2} D\ ,\ D = \frac{\pi r^2}{l^2} N \quad (9)$$

By combining equations. (1), (2), (3), (6), (7), (8), (9), the total outgoing traffic T_{all} can be written as a function of N and R as following:

$$\begin{aligned} T_{all} &= \Big(C_{cs}ND + C_a(4\frac{r}{l}N + D)(1-(1-R)^N) \\ &\quad + 4C_h\frac{FNMD}{\pi l}\Big)(1-R)^N + C_{pp}N(1-(1-R)^N) \\ &= \Big((C_{cs}\pi + 4C_h\frac{FM}{l})\frac{r^2}{l^2}N^2 \\ &\quad + C_aN\frac{r}{l}(4 + \frac{\pi r}{l})(1-(1-R)^N)\Big)(1-R)^N \\ &\quad + C_{pp}N(1-(1-R)^N) \end{aligned} \quad (10)$$

4.2 Performance Results

It was observed that average D=3.788 with r=100m, in Gayax [14] at Dec. 12. 2003 by analyzing the profile results with *gprof* after about 24 hours running of Gayax server for one day of close beta-testing phase. It was also observed that the average packet size is 25.8 bytes/packet, and the average number of packets from a client is 0.72614 packet/sec. C_{cs} and C_a have similar nature. Both constants depend on how many packets are sent by PCs in a given time. C_a can be regarded as C_{cs}. The average speed (M) and ratio of moving PCs (F) are chosen as 5m/sec and 0.5 respectively. The constant C_h is chosen as 1, because one packet is sufficient for each PC that is near from just handoff-in PC. The value of l is chosen as 400m to set the value of N in a reasonable range as discussed in section 3.2. With the parameters from Gayax server, N becomes 19.29 by eq. (9), and in plotting, the cases of N is chosen from 10 to 40 at intervals of 10.

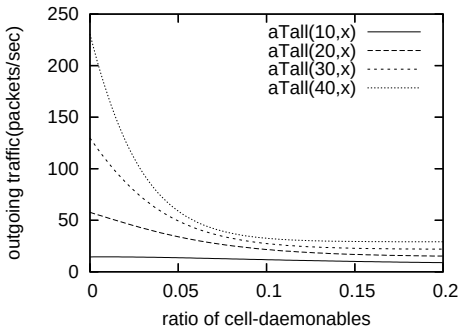


Fig. 5 outgoing traffic

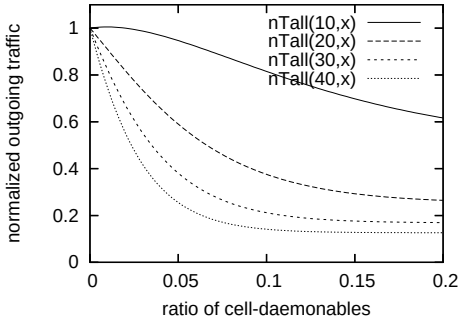


Fig. 6 normalized outgoing traffic

In eq. (10), T_{all} means outgoing traffic of CCD for a cell, and can be plotted as continuous function of R for outgoing traffic of whole cells because the virtual space consists of a number of cells. Fig. 5 and 6 show graphs of $T_{all}(N, R)$ function in several cases according to average number of PCs in a cell N, with assumption of C_{cs}=C_{pp}. In real world, C_{cs} would be much greater than C_{pp} because C_{cs} is for normal game actions and C_{pp} is for relatively rare meta-actions. It was also observed that C_{pp} is about 1% of C_{cs} at Gayax. Even if C_{pp} may be increased by adopting our HYMS architecture, the assumption of C_{cs}=C_{pp} is extremely conservative.

Fig. 5 shows outgoing traffic when C_{cs}=0.72614, and Fig. 6 shows a normalized graph. The intersections on the y-axis mean outgoing traffic of conventional client-server architecture, and it validates above modeling that observed number of outgoing packets from

Gayax is about 70 packet/sec for a cell when N=19.29. The results show that HYMS architecture reduces outgoing traffic of central server to about 60% of original amount in case of N=20, even if there is only 5% of cell-daemonable clients. The traffic reduction becomes 80% when $30 \leq N$ and $10\% \leq R$.

When N=10 and $R < 0.05$, abnormal increase of traffic is observed in Fig. 6. This is the overhead of HYMS when a master CCD sends packets to adjacent MCD. By equation (8), about $1 - (1-0.05)^{10} \simeq 40\%$ or less of all cells have MCD and the remaining 60% cells have central CCDs. This kind of ratio, i.e., similar number of MCDs and CCDs, generates much inter-cell traffic from a master CCD to adjacent MCDs, and it appears as outgoing traffic of central server. Larger percentage of either MCD or CCD will reduce such traffic significantly. However, absolute amount of this overhead is small when N is small,i.e., asymptotic amount is smaller than $O(N)$. Large N breaks rapidly the equilibrium between master CCDs and MCDs by increasing P in eq. (8), so this overhead is negligible in either case.

5. Related works

Traditional client-server based MMOGs such as EverQuest and Gayax are suffering from huge and centralized traffic on the server. Traffic distribution among multiple servers is possible by mirrored server [2], [5], generic proxy [6], [7] or booster box [8], but they cannot reduce total bandwidth-charge from IDCs. In HYMS, the server-side traffic is distributed among the clients, and thus a game publisher can save bandwidth-charge. Traffic distribution among the clients is also possible by end system multicast [3], but it is suffering from large network latency. In HYMS, there is no multi-layered hierarchy from pure clients to cell daemons, and thus the latency is smaller than end system multicast.

The techniques for MMOG servers have been matured from 90's. There are some commercial scalable and seamless MMOG middlewares: Butterfly.net[15], Zona[16], Open Skies[17], VAST[18], Terraplay[19], DEx[20], Eterna[21], Lithtech Discovery System[22], Turbine Engine[23], NeL[24] and Twisted[25]. Some of them attempt to reduce server-side bandwidth by transferring subsidiary traffic such as multimedia chatting data in peer-to-peer manners. However, the impact of the attempt is limited since the main game traffic still goes through the central servers on those systems. In HYMS architecture, the main game traffic is also distributed, and thus the server-side bandwidth can be effectively reduced.

There are several more recent proposals to apply full peer-to-peer features to MMOG server [26], and build another hybrid scheme in [27]. However, there still exists the probability of *catastrophic failures* in [26], and neither of them argues on the NAT and firewall related issues that can restrict wider accesses to a MMOG. We discussed such issues in this paper and proposed proper treatments for them.

In HYMS architecture, the client-server relation from PCs to the server is still maintained by sending every PCs' notifications to the corresponding CCD and SCDs as well the MCD. With this client-server relation, most P2P intrinsic issues such as consistency and security are solved (or masked) as described in earlier sections.

6. Conclusion and future works

In this paper, we presented a novel architecture for MMOG server. Unlike traditional client-server scheme, our HYMS architecture utilizes high power client machines as local game servers enabling clients to play game through P2P communication, and thus reduces significantly the outgoing traffic from a central game server. Since outgoing traffic is much larger than incoming traffic for MMOG and network cost is a major concern for MMOG game publishers, our scheme will help them to design games in larger scale.

We present a scheme using redundant cell-daemons for unexpected client crash, and related discussions for issues raised by adopting P2P schemes in MMOG. An analysis shows that up to 80% of server-side outgoing traffic can be eliminated even if just 10% of clients are cell-daemonable. We have a plan to simulate HYMS using real workloads from Gayax server in near future.

In HYMS, the test whether a client machine can be a MCD or not is performed with static guidelines but this test can utilizes dynamic factors like current number of players in the cell, current network load, and so on. When a client examines itself on cell-daemonability at joining, it can get answer by a number or range rather than just yes or no. With this new test, it is natural to adopt techniques such as *dynamic partitioning/grid grouping* in [28]. The cells are evenly partitioned initially, but they can change their sizes on-the-fly with those techniques. Asymmetric cell division and hexagonal cell partitioning[29] [30] which were used for load balancing need to be modified for HYMS architecture. Furthermore, since communication locality between clients and MCD in the same cell is very important for reducing network latencies, cell partition/merge methods should consider this factor.

References

[1] A. Jarett, J. Estanislao, E. Dunin, J. MacLean, B. Robbins, D. Rohrl, J. Welch, and J. Valadares, IGDA Online Games White Paper 2nd Edition – March 2003, IGDA Online Games Committee, 2003.

[2] E. Cronin, B. Filstrup, and A. Kurc, "A distributed multiplayer game server system," May 2001. EECS589 Course Project Report, University of Michigan.

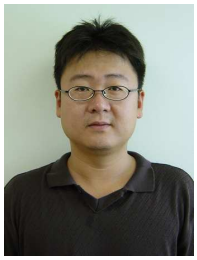
[3] S.S. Yang-hua Chu, Sanjay G. Rao and H. Zhang, "A case for end system multicast," IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking

Support for Multicast, vol.20, no.8.
[4] S. Benford, C. Greenhalgh, T. Rodden, and J. Pycock, "Collaborative virtual environments," Commun. ACM, vol.44, no.7, pp.79–85, 2001.
[5] E. Cronin, B. Filstrup, A.R. Kurc, and S. Jamin, "An efficient synchronization mechanism for mirrored game architectures," Proceedings of the 1st workshop on Network and system support for games, pp.67–73, ACM Press, 2002.
[6] M. Mauve, S. Fischer, and J. Widmer, "A generic proxy system for networked computer games," Proceedings of the 1st workshop on Network and system support for games, pp.25–28, ACM Press, 2002.
[7] C. Griwodz, "State replication for multiplayer games," Proceedings of the 1st workshop on Network and system support for games, pp.29–35, ACM Press, 2002.
[8] D. Bauer, S. Rooney, and P. Scotton, "Network infrastructure for massively distributed games," Proceedings of the 1st workshop on Network and system support for games, pp.36–43, ACM Press, 2002.
[9] "IDC service usage stipulation(in korean)." http://www.epidc.co.kr, March 2003. Enterprise Networks Inc., Seoul, Korea.
[10] J.L. Bentley, "Multidimensional binary search trees used for associative searching," Comm. ACM, no.18, pp.509–517, 1975.
[11] J. Smed, T. Kaukoranta, and H. Hakonen, "Aspects of networking in multiplayer computer games," Proceedings of International Conference on Application and Development of Computer Games in the 21st Century, ed. L.W. Sing, W.H. Man, and W. Wai, Hong Kong SAR, China, pp.74–81, Nov. 2001.
[12] "North star - broadband internet speed test." http://speed.nca.or.kr/english. National Computerization Agency, Seoul, Korea.
[13] C. Greenhalgh, "Awareness-based communication management in the MASSIVE systems," Distributed Systems Engineering, vol.5, no.3, pp.129–137, 1998.
[14] "online game GAYAX." http://www.gayax.com. Gonusoft Inc., Seoul, Korea.
[15] "Butterfly.net." http://www.butterfly.net. Butterfly.net Inc., Martinsburg, WV.
[16] "Zona." http://www.zona.net. Zona Inc., Santa Clara, CA.
[17] "Open skies." http://www.openskies.net. Cybernet Systems, Ann Arbor, Mich.
[18] "VAST." http://www.rebelarts.com. Rebel Arts, Calabasas, Calif.
[19] "Terraplay." http://www.terraplay.com. Terraplay AB, Solna, SWEDEN, +46-8-764 91 00.
[20] "DEx." http://www.horizongot.com. Horizon, a Glimpse of Tomorrow, Monrovia, Calif +1 (626) 446-8925.
[21] "Net z and eterna." http://www.quazal.com. Quazal, Montreal, PQ CANADA, +1 (514) 395-4646.
[22] "Lithtech discovery system." http://www.lithtech.com. Lithtech, Kirkland, Wash., +1 425-739-1659.
[23] "Turbine engine." http://www.turbinegames.com. Turbine Entertainment Software, Westwood, Mass, +1 781-407-4000.
[24] "NeL." http://www.nevrax.com. Nevrax, London, UK.
[25] "Twisted." http://www.twistedmatrix.com. Twisted Matrix Labs.
[26] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games." http://www.cis.upenn.edu/ hhl/Papers/infocom04.pdf, 2004. Department of Computer and Information Science, University of Pennsylvania.
[27] E. Brooks, P. Noyes, and J. Seidel, "Distributed MMOG game engine." http://www.cs.ucsb.edu/ rich/class/cs189A/winter.04/MMOG.ppt, Jan. 2004. CS189A Course Project Proposal, University of California, Santa Babara.
[28] M. Hori, T. Iseri, K. Fujikawa, S. Shimojo, and H. Miyahara, "Scalability issues of dynamic space management for multiple-server networked virtual environments," Proceedings of IEEE Pacific Rim Conf. on Communications, Computers and signal Processing, pp.200–203, 2001.
[29] S. Fiedler, M. Wallner, and M. Weber, "A communication architecture for massive multiplayer games," Proceedings of the 1st workshop on Network and system support for games, pp.14–22, ACM Press, 2002.
[30] M.R. Macedonia, D.P. Brutzman, M.J. Zyda, D.R. Pratt, P.T. Barham, J. Falby, and J. Locke, "NPSNET: a multiplayer 3d virtual environment over the internet," Proceedings of the 1995 symposium on Interactive 3D graphics, April 09-12 1995.

Acknowledgments

Special thanks to Dr. Song, M. J. who helped us rearrange and reorganize the manuscript, and the anonymous referees for their many suggestions that improved the content of this paper.

Kyoung-chul Kim received the B.S. and M.S degrees from Korea Advance Institute of Science and Technology (KAIST), in 1992 and 1994, respectively. From 1992 to 2000, he worked concurrently with Hankook Compugraphy co. located in Seoul. He is currently a Ph.D. candidate working in the computer architecture group at KAIST, and concurrently a chief programmer in Gayax server team at Gonusoft Inc. from 2001. His research interests include computer architectures, parallel processing, and MMOG server



Ikjun Yeom received his B.S. degree in Electronic Engineering from Yonsei University, Seoul, Korea in February 1995 and the M.S. and Ph.D degrees in Computer Engineering from Texas A&M University in August 1998 and May 2001, respectively. He worked at DACOM co. located in Seoul between 1995 and 1996 and at Nortel Networks in 2000. After working as a research professor at Kyungpook National University in 2001, he has been an assistant professor in the department of Computer Science at KAIST since January 2002. His research interests are in congestion control, network performance evaluation and internet QoS.

Joonwon Lee received the B.S. degree from Seoul National University, in 1983 and the M.S. and Ph.D. degrees from the College of Computing, Georgia Institute of Technology, in 1990 and 1991, respectively. From 1991 to 1992, he was with IBM research centers. He is currently a faculty member at KAIST. His research interests include computer architectures, operating systems, parallel processing, and grid & mobile computing.