July 2005

Performance Modelling of Multicast Groups for Multiplayer Games in Peer-to-Peer Networks

Zheng da Wu
Bond University, Zheng_Da_Wu@bond.edu.au

Performance Modelling of Multicast Groups for Multiplayer Games in Peer-to-Peer Networks

Z. D. Wu
School of Information Technology, Bond University, Australia
zwu@bond.edu.au

Abstract

A computer game is usually considered as a finite state machine. For a distributed multiplayer game, state information has to be exchanged among its players when they move and interact in a virtual space. If the number of players is very large, only a small subset of the game entities, controlled by their players, is interested and forms a multicast group so as to reduce the information dissemination over the network. In this paper, analytical models are proposed for evaluating the performance of multicast groups with multiplayer games in peer-to-peer networks. The system modeling is based on two stochastic processes and their possible solutions in terms of game type, player activity, and entity vision size, mode of user input processing, and grouping strategy. The cost of game operations on consumed resources is introduced to characterize the performance measure. Numerical examples are demonstrated, which provide useful insights of the behaviors of such systems.

1. Introduction

Over the last decade multiplayer computer games on the Internet, as a distinct class of distributed interactive real-time applications, have become prominent. However, gaming is a relatively neglected topic for communication research, especially in the aspect of performance modeling for various communication architectures with multiplayer games [5].

Reviewing this area, the communication architectures of networked games can be categorized into three classes: client-server architecture, peer-to-peer architecture and a hybrid of client-server and peer-to-peer architecture. For the third kind of architectures, federated peer-to-peer architecture is a typical example [6]. This paper deals with the last two kinds of architectures, since the both of them involves in multicast groups in peer-to-peer networks. In multicasting, a peer transmits packets to a multicast group identified by a multicast address. To receive packets from the multicast group, the peer must subscribe or join it.

In a multiplayer game the entities, which are controlled by players, generate state-update packets that are usually a subset of the peers. Therefore, a natural way to save bandwidth is to disseminate update packets only to those peers who are interested in them. The basic idea for implementing this approach is to divide a game or its virtual environment into "interest" multicast groups [2][6][7][8]. There are different strategies to perform such division. For examples, in "group-per-object" allocation strategy, each object has its own multicast group to which the object transmits its updates. In "group-per-region" strategy, the virtual environment is divided into regions that have their own multicast groups. A key challenge in the design of such system is how to reduce the cost of consumed resources while the performance required by the players can be also satisfied. In order to achieve this goal, it is necessary to investigate the performance behaviors of the multicast groups when a multiplayer game runs. The challenge for carrying out this work is that it deals with many factors, such as game type, pattern of players joining (or leaving) games, size of entity vision domain, mode of user input processing , and grouping strategy. In this paper, analytical models are proposed for evaluating the performance of multicast groups with multiplayer games in a peer-to-peer network. The system modeling is based on two stochastic processes and their possible solutions in terms of the factors above. The cost of game operations on consumed resources is introduced to characterize the performance measure. Numerical examples are provided for demonstrating the system performance behaviors.

The rest paper is organized as follows: Section 2 provides the background, terminology and assumption, which are needed for Section 3, where the detail of modeling and analysis for multicast groups are presented in terms of three types of games. Section 4 demonstrates the numerical results of system performance. Some related works are addressed in Section 5. Finally, conclusions are given.

2. Background, Terminology and Assumption

A game is a large state machine [3]. Figure 1 shows the game model with a peer-to-peer communication architecture, which is a targeted system for study in this paper.

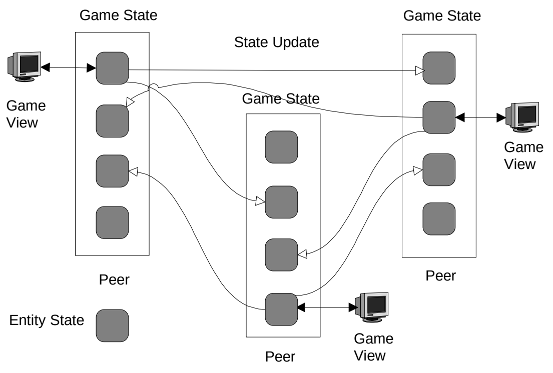

Figure 1: The game model with a peer-to-peer communication architecture

In a multiplayer game the set of information required to describe the game at a time is referred to a *game state*, which is further composed of *entities states*. An *entity* may consist of several in-game objects (e.g., military troops) and is controlled by a *player*. We may refer to a player as a person playing a game as well as the objects (e.g., avatar, tanks or monsters) that person controls in the game. Usually, one player corresponds to *a peer*. The partitioning of game states is depicted in Figure 1 also. Players make decisions, that are they decide on events that change their own states. An entity state is therefore updated based on input from the player as well as the state of other entities.

There are two approaches for processing and transmitting input events and state updates: (i) *local input processing*, where peers use the input events to update the local entity state to all other peers; (2) *distributed input processing*, in which peers transmit input events to all other peers, which then compute and correlate the entity-state update.

Entities that interact with each other form an *interaction group* or simply called as *group*. A group typically consists of a small subset of the entities in a game since the interaction only happens locally. Groups may overlap in the sense that a single entity might be part of several groups.

In a distributed multiplayer game, entities controlled by players move and interact in a two dimensional virtual space, called *play area* or *virtual environment* (VE). The location of each entity in the VE is defined by two coordinates. Suppose each entity has knowledge of its position in the VE. Each entity has a *vision domain* which describes a sub-area or the set of locations in the VE that this entity can see. In other words, each entity is only interested in receiving data from other entities within its vision domain.

The virtual environment is divided into *cells*. Thus, a vision domain will occupy a certain number of cells. Assume that each entity knows the boundaries of its vision domain and it can determine if a particular position in the VE is within its vision domain or not. Therefore, an entity has enough information about cell structure to be able determine the set of cells that intersect with its own vision domain. The assumptions above are similar to those given by Li Zou et al. in [4].

There are two basic grouping strategies as mentioned before. In the case of *cell-based* (or region-based) grouping strategy, each cell is assigned a multicast group address. This leads to two sets of cells associated with each entity, a *sending* set and a *receiving* set. As a result, each entity has two kinds of multicast groups associated with it. In the case of *entity-based* (or object-based) grouping, only a single multicast group is associated with each entity, but the cell structure for the VE is still employed. The advantage of entity-based approach over cell-based is that the amount of information messages that need to be filtered out by the recipients can be reduced [4]. Therefore, in this paper the entity-based grouping is chosen for study in more detail.

Consider entity A and its associated multicast group, denoted as G_A. An entity in G_A multicasts all its information to the members in G_A. In order to receive relevant information an entity needs to join all groups corresponding to entities within its vision domain. This is illustrated in Figure 2, in which there are five entities in a part of the VE and they form five entity-based multicast groups, denoted as G_i, $i \in \{A, B, C, D, E\}$. For examples, entity A needs to join groups G_D and $G_{E,}$ while entity E also joins group G_A. But entity B is not belonging to G_A.

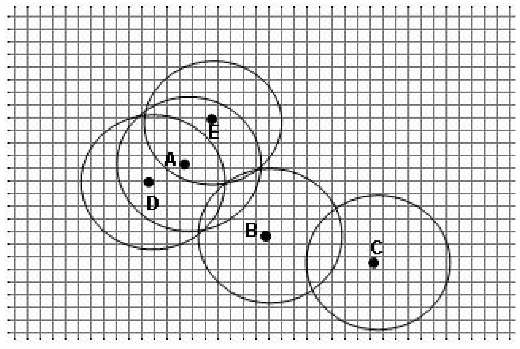

Figure 2: Cells and vision domains

A mechanism has to be employed for maintaining the groups, as the members of each group will be dynamically changed during the game runs with multiple players. Usually, this can be done by using the cell-structure of the VE and exchanging the entities positions among them [6]-[8]. Thus, we assume this mechanism is given.

In order to carry out performance evaluation of the targeted system, the performance measure should be defined in the context of consumed resources by the system. This is based on "Network Virtual Environment Information Principle" in [2] by Singal and Zyda. It is simply expressed in *information principle equation* as follows:

$$Resources = M \cdot H \cdot B \cdot T \cdot P \qquad (1)$$

where M is the number of messages transmitted, H the average number of destination nodes (or peers) for each message, B the average amount of network bandwidth required for a message to each destination, T timeliness with which the network must deliver messages to each destination, and P the number of processor cycles required to receive and process each message. The amount of consumed resources can be mapped to the cost of the system. The good example for this transformation is given in [5]. This technique is also used in the system performance analysis in next section.

As the cost of consumed resources depends on many factors, the key work for modeling the system is to find out the cost function in terms of relevant system characteristics, including game type, pattern of players join/leaving games, size of vision domains, mode of user input processing, and grouping strategy.

3. Modeling and Analysis

Suppose that a multiplayer game runs in the area of a VE, in which the maximum number of entities the game supports is N. Typically, N entities may be controlled by N players respectively. This assumption is based on the fact that any game can only supports a limited number of players even this number is very large. Each entity has a vision domain centered at the location of the entity. Its size is defined by the radius of the vision domain, denoted as γ_i, $i = 1,2,..N$, assuming the vision domain is a circle in shape as shown in Figure 2. The radius of a vision domain may vary since it depends on the type of a game. For simplicity, we assume all the entities in the VE have the same vision size, $\gamma_i = r$, and r is given. Thus, we will not model the impact of vision sizes on the system performance in this paper. When a player logs in a game, the player generates and controls his/her entity in the VE. During the game running, an entity, named as E, moves around in the area of the VE randomly. If we trace *E's* mobility relevant to other entities, two stochastic processes can be identified and denoted as $\Im\{G_E(t), t \geq 0\}$ and $\Re\{H_E(t), t \geq 0\}$, where $G_E(t)$ represents the number of entities within *E's* vision domain and $H_E(t)$ the number of other vision domains in which E resides at time t, $0 \leq G_E(t) \leq N$, $0 \leq H_E(t) \leq N$. Recall the system description in the last section, the two stochastic processes characterize the behaviors of entity E in respect to interaction with other entities, since $G_E(t)$ and $H_E(t)$ gives the information about the interaction groups associated with entity E.

Consider process $\Im\{G_E(t), t \geq 0\}$. It may be specified as a *birth-death* process. Each time when an entity enters *E's* vision domain from the rest part of the VE, it will gets a "service". This means that some interaction between entity E and others occurs in the domain. After that, it leaves the domain or service. Process $\Re\{H_E(t), t \geq 0\}$ is of the similar nature.

There are many different types of games today. This paper only deals with thee types of multiplayer games running in the VE and involving multicasting groups.

3.1 Modeling of Game Type 1

Suppose a set of games G_1 shares the following features: (i) it is allowed for any n players within the VE to play a game at a time, where $n \leq N$; (ii) the players have to take action on the game in turns once they form an interaction group; It means that when a player is *served* (say, fighting with his/her target) any other player's action will be blocked until its turn; (iii) the game service discipline is the first come first service (FCFS) in the context of a vision domain.

For a game of G_1, stochastic process $\Im\{G_E(t), t \geq 0\}$ may be further specified as a birth-death queuing system. The queuing center, abstracting the vision domain of entity *E,* has a finite population of possible entities (controlled by their players) *n*, where $n \leq N$. Any entity in the VE has two states, either in *E*'s vision domain (waiting or taking action) or outside of the vision domain in some sense *arriving*. It means that entity *E* may return to service again after it finishes service. Assuming the arrival time and service time of an entity are two random variables with exponential distributions whose means are $1/\lambda$ and $1/\mu$ respectively. All entities in the VE act independently of each other. As a result, when there are *k* entities in the domain then there are *(n-k)* entities in the arriving state. Therefore, the total arrival rate in this state is $\lambda(n-k)$. This system is in a strong sense self-regulating. It means that when the system gets busy, with many entities in the domain, then the rate at which additional entities arrive is in fact reduced, and thus lowering the further congestion of the system. This is called *M/M/1/N* queuing model in [1], where

$$\lambda_k = \begin{cases} \lambda(N-k), & 0 \leq k \leq N \\ 0 & , \textit{otherwise} \end{cases} \tag{2}$$

$$\text{and} \qquad \mu_k = \mu, \quad k = 1, 2, ...N$$

Thus, we can obtain the distribution of stationary probabilities of finding the system with *k* entities, p_k, as below

$$p_k = \begin{cases} p_0(\lambda/\mu)^k \dfrac{N!}{(N-k)!}, & 0 \leq k \leq N \\ 0, & k > N \end{cases} \tag{3}$$

where

$$p_0 = \left[\sum\nolimits_{k=0}^{N} (\lambda/\mu)^k \frac{N!}{(N-k)!} \right]^{-1} \tag{4}$$

As a result, the expected number of entities in the system with a game of G_1 is given by

$$g_1 = \sum\nolimits_{k=0}^{\infty} k \cdot p_k \tag{5}$$

In order to evaluate the cost of consumed resources we need to compute the average rate, at which g_1 changes in *E*'s vision domain due to entities leaving or joining. This change reflects that the entities are in a state of switching their interact groups. Thus, this rate is called the average *switch rate*, denoted as ω_1 for G_1 games, and it can be calculated by

$$\omega_1 = |\bar{\lambda} - \bar{\mu}| \tag{6}$$

where $\bar{\lambda}$ and $\bar{\mu}$ represent the average arrival rate and the average service rate for total $N+1$ entities, including zero entity, as a result,

$$\omega_1 = |(N+1)^{-1} \sum_{k=0}^{N} \lambda(N-k) - \mu| \tag{7}$$

3.2 Modeling of Game Type 2

Suppose a set of games G_2 has the following common features: (i) it is the same as given in (i) of the features of G_1; (ii) once a player's entity enters *E*'s vision domain, the action taken by the player can be immediately served in the game without any waiting and blocking; (iii) any *n* players can play the game in parallel way when they reside in the domain, where $n \leq N$; Thus, the FCFS discipline is not applied in this case. As a result, G_2 may be modeled as *M/M/*∞*/N* queuing system, where λ_k is defined as the same as in equation (2), but $\mu_k = k \cdot \mu$, *k =1, 2,…, N*. Clearly, this is an ergodic system too [1]. After solving this system, we can obtain the distribution of stationary probabilities of finding the system with *k* entities, p_k, for G_2 as below:

$$p_k = \begin{cases} \dfrac{(\lambda/\mu)^k \binom{N}{k}}{(1+\lambda/\mu)^N}, & 0 \leq k \leq N \\ 0, & \textit{otherwise} \end{cases} \tag{8}$$

The expected number of entities in the system with a game of G_2 can be computed by

$$g_2 = \sum_{k=0}^{N} k \cdot p_k = \frac{N\lambda/\mu}{1+\lambda/\mu} \tag{9}$$

Based on the same argument about the average switch rate ω_1 given in the last subsection, the average switch rate for a G_2 game, ω_2, is calculated by

$$\omega_2 = (N+1)^{-1} \left| \sum_{k=0}^{N} (\lambda(N-k) - k\mu) \right| \qquad (10)$$

3.3. Modeling of Game Type 3

Third type of games G_3 is different from the last two types is as follows: (i) there is a *threshold* controlling the number of players with a game of G_3; suppose the value of threshold is *K*, where $0 \le n \le K \le N$ and *n* represents the number of players allowed to play the game. This assumption is based on the fact that many games only support a certain number of users to interact at a time; Thus, if entities arriving to find *K* already in the system are *lost* and returns immediately to the arriving state as if they had just completed services. (ii) These *n* players can be served in a parallel way, but is limited to *m*, where $0 \le m \le K$. These features lead to a birth-death process, called *M/M/m/K/N* model [1], where λ_k is defined as the same as in equation (2), but

$$\mu_k = \begin{cases} k\mu, & 0 \le k \le m \\ m\mu, & k \ge m \end{cases}$$

The distribution of stationary probabilities of finding the system with *k* entities, p_k, for the case of $N > K = m$ with a G_3 game can be given by

$$p_k = \frac{\binom{N}{k}(\lambda/\mu)^k}{\sum_{i=0}^{m}\binom{N}{i}(\lambda/\mu)^i} \qquad (11)$$

Thus, the average number of entities in the vision domain of entity *E* for G_3 is given by

$$g_3 = \sum_{k=0}^{N} k \cdot p_k$$

Similarly to computing ω_1 and ω_2, we can calculate the average switch rate, ω_3, for game type G_3, as below

$$\omega_3 = N^{-1} \left| \sum_{k=0}^{N-1} \lambda(N-k) - \sum_{k=0}^{m} k\mu - (N-m)m\mu \right| \qquad (12)$$

3.4 Analysis

Let us analyze second stochastic process $\Re\{H_E(t), t \ge 0\}$ now. It characterizes how many interaction groups entity *E* joins at time *t*. The idea for analyzing this process is based on the relation of $G_E(t)$ and $H_E(t)$.

Before this relation is given, let us verify the following fact at first: (i) if entity *x* is within the vision domain of *y*, then entity *y* must be inside of the vision domain of *x*, for example, entities *A* and *E* satisfy this relation shown in Figure 2; (ii) if entity *x* is not within the vision domain of *y*, then entity *y* must be outside of the vision domain of *x*, for example, entities B and C demonstrate this situation in Figure 2. This fact is subjected to the condition that all possible entities *N* in the VE must have the same size of vision domains. This fact leads to a special relation of $G_E(t)$ and $H_E(t)$, namely, $G_E(t) = H_E(t)$. As a result, their average values satisfy the equation $g_i = h_i$, when the system is in equilibrium, for all game type $G_i, i = 1, 2, 3$.

In contrast, if there are two or more entities with different vision sizes, the relations $G_E(t) = H_E(t)$ and $g_i = h_i$ will be violated.

3.5 Cost Computation

There are two types of resources consumed by games: (i) internal processing resource deals with handling input from players, correlating the entity states and visualize the states of entities to the players; (ii) network processing resource involving in reception and transmission of user-input events and state-update messages. The cost of operations on resources is well introduced by Bauer et al. in [5] for multiplayer games. Their definitions are employed in this paper as follows:

$p_i(x)$ - The cost of processing *x* user-input event; $p_c(x)$ - the cost of correlating *x* entity states of an interactive group; $t_i(x)$ - the cost of transmitting a user-input event to *x* destinations; $t_u(x)$ - the cost of transmitting an entity-state update to *x* destinations; $r_i(x)$ - the cost of receiving *x* user-input events; $r_u(x)$ - the cost of receiving *x* entity-state updates.

As a result, the average cost for internal processing C_i is generally expressed by

$$C_i = p_i(x_A) + B_A p_c(g_A) \qquad (15)$$

where the number x_A of the user-input events, the constant B_A, and the average size g_A of the interaction group depend on the game communication architecture chosen. Similarly, the average cost for network processing C_n is given by

$$C_n = t_i(x_{A1}) + r_i(x_{A2}) + t_u(x_{A3}) + r_u(x_{A4}) \qquad (16)$$

where the variables x_{Aj}, j = 1, 2, 3, 4, depend on the game architecture considered.

As discussed before, there are two modes for the user-input processing. If the local input processing is employed with each peer, the average input processing cost C^l can be evaluated by using equations (15) and (16) as follows:

$$C_n^l = t_u(g-1) + r_u(g-1)h$$
$$C_i^l = p_i(1) + p_c(g)h$$
$$C^l = C_i^l + C_n^l$$

where C_i^l and C_n^l represent a peer's input and network processing cost respectively; *g* the average number of entities of an interaction group, specifically equal to g_i, *i*=1, 2, 3, as analyzed in the last section; *h* the average number of interaction groups that a player (or entity) joins, which corresponding to $h_i, i = 1,2,3$, as given before. If distributed input processing is used for each peer, the average input processing cost is determined as below:

$$C_n^d = (t_i(g-1) + r_u(g-1))h$$
$$C_i^d = (p_i(g) + p_c(g))h$$
$$C^d = C_i^d + C_n^d$$

To compute the cost for maintaining the interaction groups, the average switch rate ω corresponding to ω_i, i = 1, 2, 3, should be considered, since they characterize the dynamic behaviors of the multicast groups. The average maintenance cost for a multicast group in a peer-to-peer network can be calculated by the following equations:

$$C_n^m = d(t_u(g) + (\frac{N \cdot h}{g}) + r_u(g)h$$
$$C_i^m = d \cdot p_c(g)h$$
$$C^m = C_i^m + C_n^m$$

where C_i^m and C_n^m represent input and network processing cost for maintenance of multicast groups respectively. Finally, the total cost of the targeted system is given by

$$C = C^m + C^l \ or \quad C^m + C^d \qquad (17)$$

4. Numerical Results

Before evaluating the system cost, the cost of operations on resources, defined in section 3.5, needs to be specified. In our numerical analysis, we define $p_x(x)$, $r_i(x)$, $r_u(x)$, $t_i(x)$, and $t_u(x)$ as linear functions, namely, $p_x(x) = \delta_i \cdot x$, $r_i(x) = \theta_i \cdot x$, $r_u(x) = \theta_u \cdot x$, $t_i(x) = \theta_i \cdot x$ and $t_u(x) = \theta_u \cdot x$; In addition, $p_c(x)$ is expressed as $p_c(x) = [x(x-1)/2] \cdot \delta_c$, where the term in brackets is the number of all entities pairs to be correlated within an interaction group of size *x* [5]. Note that the factors δ_i, δ_c, θ_i and θ_u are game dependent.

Four figures will be demonstrated as examples of numerical results. During computing the system cost *C*, we initiate $\delta_i = \theta_i = 1$, $\theta_u = 3$, and $\delta_c = 2$. Also, the distributed input processing mode is chosen for the analysis in these examples as it leads to more communication overhead comparing with local one.

Figure 3 shows the system cost *C* via the ratio of the average arrival rate to the average service rate, $\rho = \lambda / \mu$, based on Game Model 1. $C\ (N = i\ )$ represents the cost evaluated with total possible entities N in the VE, where $N = 20, 40, 60$; It notes that there is a special point, at which the system cost is of the minimum value once all input parameters are fixed; This is because when the system average arrival rate and service rate approaches to a balance, the average switch rates ω_1 will have a minimal value closed to zero.

Figure 4 presents the system cost visa ρ based on Game Model 2. In this type of games, a player can have a service immediately without any blocking when she/he joins the system. Thus, the cost is increased together with ρ getting large. This is reasonable as more number of entities in the system more user input and network processing are required.

Figure 5 demonstrates the performance for Game Type 3. As the system has a threshold for controlling group size, the cost quite depends on threshold value *m*. If all other input parameters are fixed, naturally, smaller *m* leads to lower cost.

Figure 6 gives the performance comparison among the three types of games, where the cost of game type 2 is increased more rapidly when ρ is getting large, comparing with the other two types of games. The reason is that game type 1 has a rule in which only one player is served at a time while game type 3 has a threshold for controlling group size. But, game type 2 does not have any constraint at all.

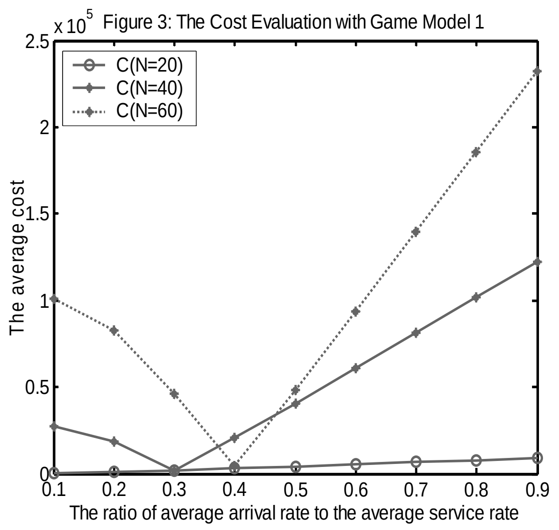

Figure 3: The Cost Evaluation with Game Model 1

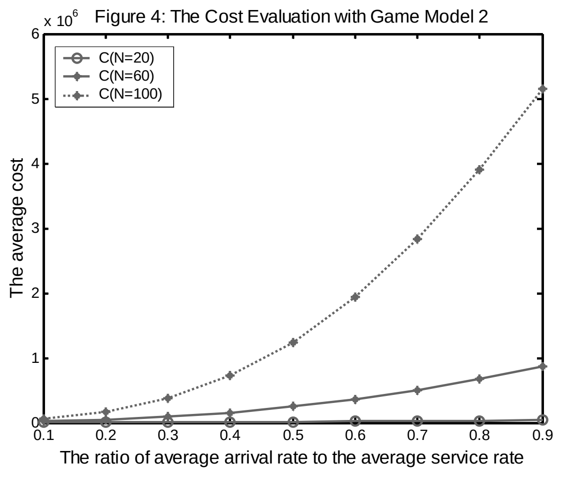

Figure 4: The Cost Evaluation with Game Model 2

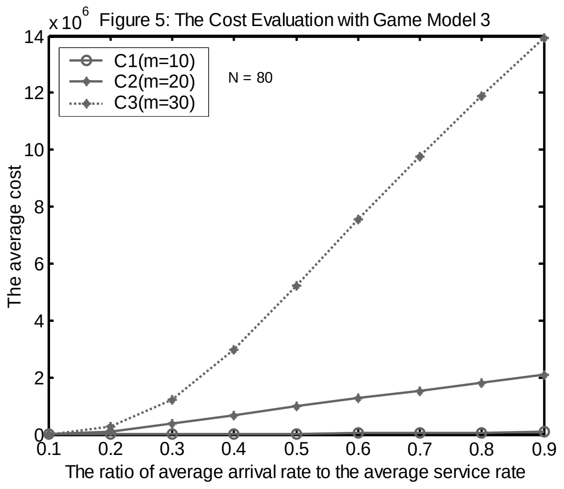

Figure 5: The Cost Evaluation with Game Model 3

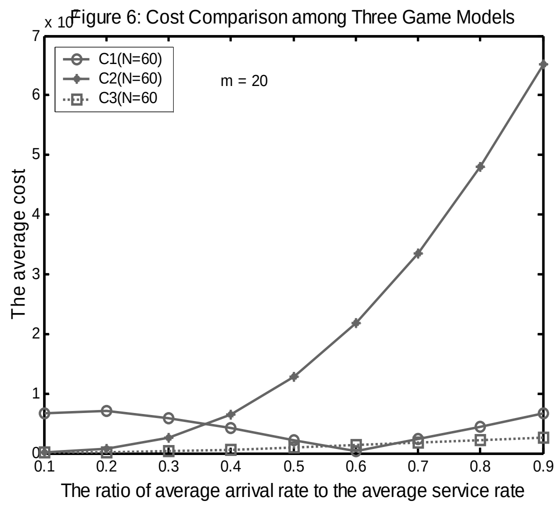

Figure 6: Cost Comparison among Three Game Models

5. Related work

N. E. Baughman et al. [3] present a set of diagrams to model client-server and distributed (or peer-to-peer) communication architectures for multiplayer games in order to explore exploits possible for cheating in such systems. The model gives a method for describing a distributed game as a large state machine. Figure 1 in the current paper is originally based on this model.

In paper [4] by Li Zou et al, the authors propose their static and dynamic simulation models for the performance evaluation of game state dissemination with cell-based and entity-based strategies. In the paper the basic concepts for distributed games, such as entity, player, cell, vision domain, player area and interaction group, are well defined. The difference from their work the current paper gives analytical models in terms of different types of games and other factors.

Daniel Bauer et al. [5] proposed a very general model for evaluating the scalability of massive multi-player games with three communication architectures: client-server, peer-to-peer and federation peer-to-peer. They define the cost of operations on input processing resource and network processing resource. This is the first paper gives a quantitative assessment of these architecture by using analytical models. The difference from the current paper, they assume the expected values of three random variables, g, h and ω , are given before the evaluation and then use them as input parameters for their cost analysis directly.

Jouni Smed et al. [2] give an excellent review on networking and multiplayer computer games. The author of the current paper benefits their work from the conceptual view of multiplayer games. Their discussion about the challenges in this area is also helpful.

Conclusions

This paper presents a preliminary work on the performance modeling of multicast groups for multiplayer games in peer-to-peer networks by using stochastic processes. The analytical models, based on Markovian queuing systems, have been developed which provide useful insights into the system behaviors in terms of relevant characteristics, game type, pattern of players activity, vision domain size, mode of input processing, and group strategy. Work is in hand to further refine the models in order to correlate them with operational multiplayer games on the networks of today.

References

[1] Leonard Kleinrok, “Queuing Systems, Volume 1, Theory”, Wiley, 1975.

[2] Jouni Smed, Timo Kaukoranta, and Harri Hakonen, “A Review on Networking and Multiplayer Computer Games”, Turku Centre for Computer Science, Technical Report No 454, April 2002.

[3] Nathaniel E. Baughman and Brian Neil Levine, “Cheat-Proof Playout for Centralized and Distributed Online Games”, Proc. of IEEE INFOCOM 2001, pp.104-113.

[4] Li Zou, Mostafa H. Ammar and Christophe Diot, “An Evaluation of Grouping Techniques for State Dissemination in Networked Multi-User Games”, Proc. of 9th International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication System (MASCOTS), pp.33-40.

[5] Daniel Bauer, Ilias Iliadis, Sean Rooney and Paolo Scotton, “Communication Architectures for Massive Multiplayer Games”, Multimedia Tools and Applications, 23, pp. 47-66, 2004, Kluwer Academic Publishers.

[6] Sean Rooney, Daniel Bauer, and Rudy Deydier, “A Federation Peer-to-Peer Network Game Architecture”, Research Report, IBM Zurich Research Laboratory, RZ 3528 (#99542), 2004.

[7] B. Knutsson, H. Lu, W. Xu and B. Hopkins, “Peer-to-Peer Support for Massively Multiplayer Games”, Proc. of IEEE INFOCOM 2004.

[8] S. Fiedler, M. Wallner, amd M. Weber, “A Communication Architecture for Massive Multiplayer Games”, ACM Proc. of NetGames 2002, pp.14-22.

[9] J. D. Pellegrino and C. Dovrolis, “Bandwidth Requirement and State Consistensy in three Multiplayer Game Architectures”, ACM Proc. of NetGames 2003, pp.52-59.