Draft

Monday 10th May, 2010

1 Design

1.1 The flow

1.1.1 Basic idea

We aim to use P2P technology to distribute the information (i.e. game state messages and neighbor information) in the MMOGs, where each peer sends to and receives from other peers whose avatars are nearby in the virtual game world. Thanks to the locality of peers' interests, they only need to know what happens nearby. Peers can receive the game state messages from these peers directly can play their avatars, to alleviate the server. Nevertheless, the popularity of the regions in the game world is highly skewed [1], and some peers in the "hot" regions (i.e. where many avatars are there) need to disseminate much more messages to neighbors than others in the system. To make best use of peers' resources, we have peers with extra upload capacities help those in hot regions, in return, they will be able to receive the help from others when they are in the hot regions, due to the avatar movement in the game. In our design, the social friendship is also considered in the system, i.e., friends who have stronger relation may be more likely to help each other.

1.1.2 Incentives

We observe that the incentive mechanism here is quite different from that in Bittorrent systems, e.g., the trading between two peers is not happening at the same time (i.e., a peer might need to help another peer even if it can't get the help back immediately). In our design, receipts and game points are traded among friends and strangers respectively, to achieve such indirect reciprocity in the system.

We amount of receipts or game points are evaluated by the contribution a peer has done to others: (1) the assistance that peers help friends to relay their game state messages; (2) the share of neighbor information, i.e., a peer can ask a friend or a stranger for the neighbor information. In our design, we give different incentive mechanisms to that between friends and among strangers, and also a combination strategy for the two different mechanisms.

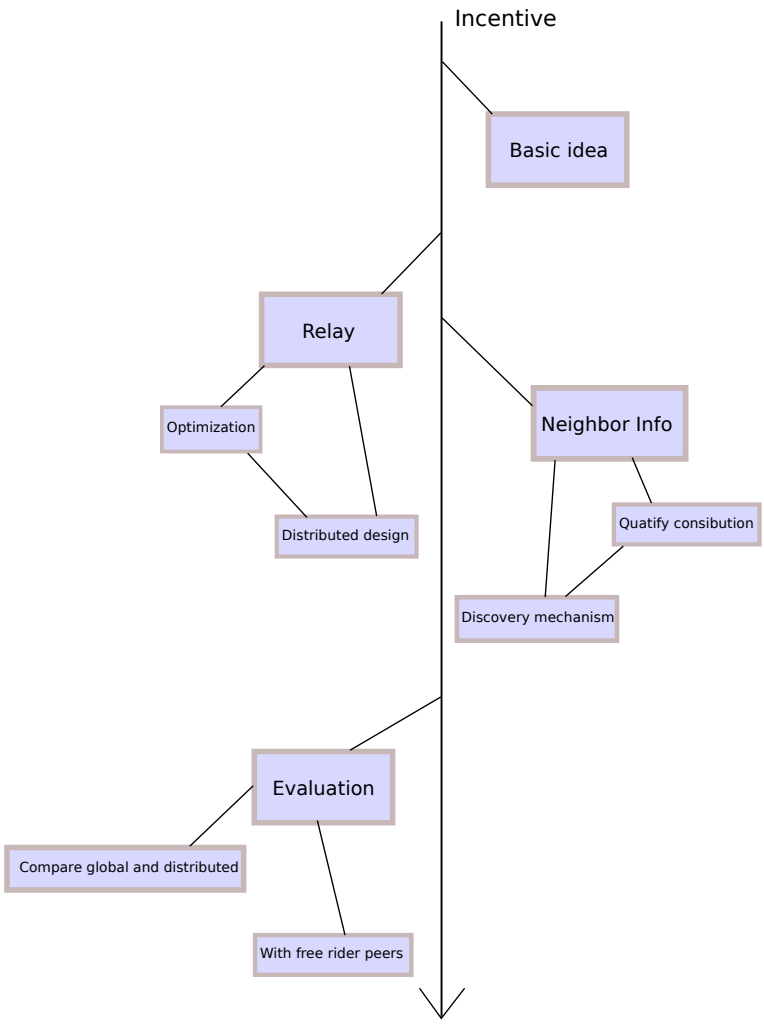

Figure 1: The flow

2 Incentives

2.1 Participants in the Game

- Friend-Friend: when a peer wants some help (relaying or neighbor information) from other peers, it is easier to get the help from a friend because they have close bounder in the social network, so that a friend can offer the help without having being helped before.

- Stranger-Stranger: for strangers, there might be two approaches for one to help the other for the first time: (1) find a friend chain which connects the two strangers by media intermediate friends, which are the "warrantors" to make sure that the helper is able to get the contribution back some time; (2) use global currency to pay for the help, so that the helper can later get help by the credits. In our design, for strangers, we use the later approach for incentives among strangers, and the credits are the game points which are commonly used in the online games.

2.2 Contributions in the Design

Both relaying game state messages or sending neighboring information to other peers are the contributions a peer can make to the system. A peer in some hot regions might need other peers to help it distribute its game state messages to all the neighbors, on the other hand, it has much more chances to send other peers the necessary neighbor information for them to find neighbors. So a peer who always stays in hot regions or cold regions are both able to contribute to the system, by relaying and sending neighboring information, respectively.

In our design, we try to find the best quantization scheme for the contribution, so that peers in the system can get best viewing quality without impairing the fairness among peers.

- Relay: (1) upload resource; (2) latency

- Neighbor information

2.3 Quantify the contribution

2.3.1 Contribution types

A peer can contribute to other peers by either relaying game state messages for them or giving neighbor information to them. So total contribution is the combination of the two contributions:

- Relay: what impact the contribution of relay? (1) the bit rate the game state messages r_g; (2) the promised time to distribute the messages t_r. The contribution by relay CR that peer i has done for peer j then can be calculated:

$$CR_{ij} = a_1 \cdot \log(r_g) \cdot T_r$$

T_r is the time each relay takes, i.e., i promises to help for T_r seconds.

- Neighboring information: the following metrics impacts the neighbor information contribution that peer i has done for peer j, the weight of the neighbor w_n;

$$CN_{ij} = a_2 \cdot w_{nj}$$

The importance indicator w_{nj} of a neighbor to peer j is evaluated by the distance $dis(j, n)$ between j and n in the virtual game world:

$$w_{nj} = \frac{1}{\log(dis(j, n))}$$

2.3.2 Quantify the contribution among friends

We use the matrix $\{f_{ij}\}_{N \times N}$ to denote the social relationship between peers in the system. The larger f_{ij} is, the closer the two peers are to each other. For strangers, the f_{ij} value is much larger than that of friends.

a_1 and a_2 are the parameters to adjust the weights for the two contributions. We have the total contribution that a peer i has done for peer j :

$$C_{ij} = CR_{ij} + CN_{ij}$$

2.3.3 Quantify the contribution among strangers

Game points are used for a peer to "buy" some contribution from strangers. The number of game points that is charged also depends on the contribution C_{ij}. Here we use a simple transfer that the number of game points to be charged by peer i from peer j is:

$$GP_{ij} = \frac{\alpha}{f_{ij}} \cdot C_{ij}$$

α is a exchange parameter to convert the contribution value to the game points. How to exchange the game points gained from strangers and the contribution gained from friends. Because a peer sometimes might need help from Strangers when its friends are not able to provide the help, although it has contributed to the friends.

Table 1: Transactions

	Contracts	Game Points
Friends	Update CC_{ij} if balance holds	Transfer GP_{ij} ($B_i = B_i + GP_{ij}$,$B_j = B_j - GP_{ij}$) if $\lvert CC_{ij} - CC_{ji}\rvert > \beta \cdot f_{ij}$
Strangers		Transfer GP_{ij}
C/S (backup mechanism)		Transfer GP_{ij}

2.4 Balance in the system

We use the CC_{ij} to denote the total relay and neighboring information contribution that peer i provides to peer j, where i and j are friends, and B_i to denote the balance of peer i's game points.

Balance between friends: To keep the balance, the contribution peer i do for j should be similar to that j has done for i, and the difference should not exceed some value, which is related with the closeness of the two peers:

$$|CC_{ij} - CC_{ji}| < \beta \cdot f_{ij}$$

Here *beta* is a parameter to transfer the entry in the closeness matrix $\{f_{ij}\}_{N\times N}$ to the balance value.

No-bankruptcy balance of game points: For game points, we simply require that peers' balance of game points B_i greater than 0. This rationale ties that in most MMOGs, players are demanded to buy game points to play the games, and they always need to charge the account when the game points are used up.

$$B_i \geq 0, \quad i = 1, 2, \ldots, N$$

2.5 How to combine the two contribution

The problem is to choose a proper pair of parameters (a_1, a_2)

3 Relay Design

(TODO remove this), structure of this section: (1) basic idea; (2) mechanism; (3) optimization at individual peers

Why we need relay design?

(A) Upload resource: Since the popularity of the game regions is highly skewed, a few regions have much more peers in them than those in other regions. On the other hand, a peer always needs to receive all game state messages from all the peers nearby in its AOI. When in the hot regions, the peer may not be able to get the game state messages from some interested peers, due to the lack of upload capacities among these peers. At this moment, the peer can ask its

friends or other helper peers to join the stream overlays to help relay the game state messages, so that the peer can receive indirectly from the relay helper peers.

(B) Latency concern: Latency is one of the most important factors that impact the user experience in MMOGs (TODO:ref). Thanks to the prevalence of triangle inequality of the end-to-end latencies, it is possible for the peers to make use of their friends' relaying to reduce the latencies of receiving the wanted game state messages.

Incentives

The incentive for relay peers to help lies: (1) a peer is likely to help its friends in the system, since it trusts the friends, and sometime its friends will help it back; (2) credits could be paid to get the help, and the money can be then used to "buy" help from stranger relay peers.

Credits in the game: in our design, we use game points for incentive, as many online games require players to buy the game points to join the game. A part of the money earned by selling game points is spent on the operation of the game, so if a peer can help more other peers, it is reasonable to give it some "bonus" game points, since it help to alleviate the server cost by contributing resource to the system.

Structure of this section

In this section, we first formulate the problem to a resource allocation schedule with social network relationship of peers, and then we give an optimization framework to solve the problem, including both global optimization and distributed algorithm.

3.1 Relay Problem

In this subsection, we describe the relay problem, including the system's view and an individual peer's view. **(1) For the whole system**, we need to schedule the peers to receive and relay the different streams (for different peers, the bitrates are different) strategically to improve the stream quality for all the peers; **(2) For an individual peer**, it needs to maximize the number of streams received from interested peers and the minimize the latencies, and keep debt low with the friends.

3.2 Problem Formulation

We still use a directed graph $G_n = (R_n, N, E)$ to represent a stream overlay, where n is the source peer, and R_n is the set of peers who are to receive the stream. N is the set of all peers in the system, which can be used as relays, and E is the application-layer connections between peers.

For each stream in the system whose source is peer n, peers in R_n will try to receive the stream, whose bit rate is r_n. When a peer enters the overlay, it first tries to receive from the existing peers if they have enough upload capacities to send the stream; or it will brings in a new relay peer to receive the stream.

3.2.1 How a requesting peer schedules

How a peer schedules: when a peer moves into a new region, it will find the neighbors close to it by the peer discovery mechanism (we will discuss in Sec. **??**, and try to receive messages from the ones in its AOI.

S_n is the set of peers whose streams should be received by peer n, and $S1_n$ the set of peers whose streams are not address by peer n, i.e., not received due to the lack of upload resource or large latency. F_n is the set of friends and other peers that peer n can make use of, to get the un-addressed streams in $S1_n$.

In our design, the peer will try to schedule the relay peers in F_n to minimize the number of unaddressed streams. Let's first list the constraints: a_{in}^j is the net upload allocation of peer i for peer n for relaying stream j, i.e., the upload resource the peer i contributes minus the download resource it consumes, since it first needs to receive and then relay it.

The price of a friend's contribution

The calculation of the price is given in Sec. 2.3. In the relay problem, since only relay contribution is considered, we have

$$\begin{cases} C_{ij} = CR_{ij} & friendcontract \\ GP_{ij} = \frac{\alpha}{f_{ij}} \cdot CR_{ij} & gamepoints \end{cases}$$

CC_{ij} is the total contribution peer i has done for peer j and B_i is still the game point balance for peer i.

- **Upload constraint**:

$$\sum_{j \in S1_n} a_{in}^j \leq u_i, \quad i \in F_n$$

The upload bandwidth a

- **Balance constraint**:

$$\begin{cases} |(CC_{in} + CR_{in}) - CC_{ni}| < \beta \cdot f_{in} & i \text{ and } n \text{ are friends} \\ B_n - GP_{in} > 0 & i \text{ and } n \text{ are not friends} \end{cases}$$

Objective function

For the individual peer n, the objective is to minimized the number of unaddressed streams.

$$\max \sum_j z(j), \quad j \in S1_n$$

$$z(j) = \begin{cases} 1 & \sum_i a_{in}^j > r_j, \text{for upload constraint} \\ 1 & \sum_i a_{in}^j \geq 0, \text{for latency constraint} \\ 0 & otherwise \end{cases}$$

3.2.2 How the service peer responds

(TODO: remove this) this subsubsection is going to formulate the problem that how a peer decides which requests to serve

For a peer, deciding which requests to serve when the upload capacity is not sufficient to supply all of them, is also an optimization problem.

- Upload constrains:

$$\sum_i q_i \cdot r_{Src(i)} \cdot LI(i) < u$$

  u is the available upload capacity of the service peer. Q is the set of K requests received $\{(Peer(i), Src(i), LI(i))\}_K$, where $Peer(i)$ is the requesting peer, $Src(i)$ is the streaming source is j, and $LI(i) = 2, 1$ indicates whether for upload resource or latency. We will try to assign 0 or 1 (serve and not server) to each q_i, which is an indicator whether serves the ith request. (*problem: some requests in Q might have a same source peer*)

- Objective:
  - Maximize the number of streams (Knapsack problem)?

$$\max \sum_i q_i$$

  - Maximize the game points?
  - Owe friends least?

3.3 How a peer makes decision

this subsection describes how a peer makes the decisions according to the optimization result

3.3.1 For a requesting peer

For a requesting peer n, by running the optimal schedule, it will find the best way to put peers in F_n. The optimal objective is always to achieve a best streaming quality (i.e., to receive as much streams from peers as possible).

When some requests are not served?

3.3.2 For a service peer

1. A service peer can refuse some request helps?

2. Which strategy to apply to serve these requests, (1) best effort (maximize the number of streams being helped); (2) earn maximal game points; (3) owe friends least

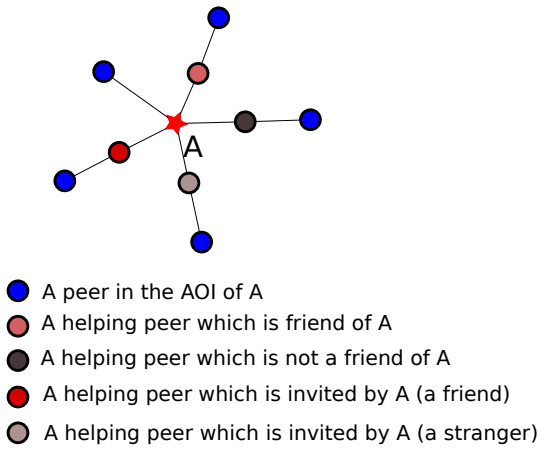

Figure 2: Pull framework

3.3.3 For the source server, i.e., how to insert the new relay peer

3.4 Optimization Framework

(TODO: remove this) merge with the formulation

A peer moves in the virtual world, and decides which neighbors' game states should be received. Only the interested neighbor's game states should be received, i.e., the peer is staring at at aiming at, while other neighbors' avatars, whom the peer is not interested in, could be guided by AI algorithms. The peer sends a control message to the neighbors that it is interested in, who then will add this peer to a receiver list. Peers need to distribute its game state messages to these neighbors in the receiver list.

More importantly, a peer needs to receive messages from the neighbors, in some modern online MMOGs, the peer is forced to go back several frames ago if some of the neighbors' game state messages are found not received. Thus, receiving all the messages from the neighbors is a big incentive for peers, since this can improve the fluency of the game controls.

In our design, peers actively pull the game state messages from the neighbors in its AOI, which are maintained by the neighbor discovery mechanism we will discuss in Sec.**??**. For the neighbors whose game state messages should be received, (1) the peer asks the neighbor to send the game state messages directly, and it will get the stream directly from the neighbor who has enough upload capacity to send one more stream; (2) or the peer will some relay peers for the neighbor if its upload capacity is exceeded, then the peer tries these relay peers to get the stream from the neighbor; (3) when the relay peers are still unavailable, the peer will invites a peer to relay for the neighbor by inserting it into an existing distribution path of the neighbor.

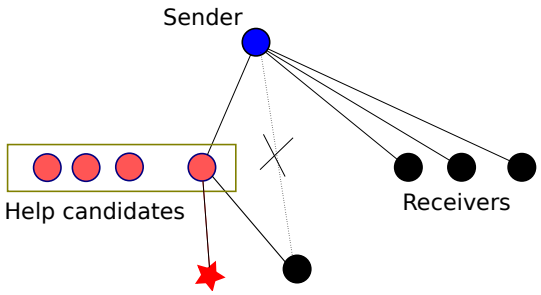

Figure 3: When the peer A finds the neighbor (and the relay peers for it) are not able to send the game state messages, it invites a relay peer from its relay candidate pool. An existing link has to be broken to insert the new relay

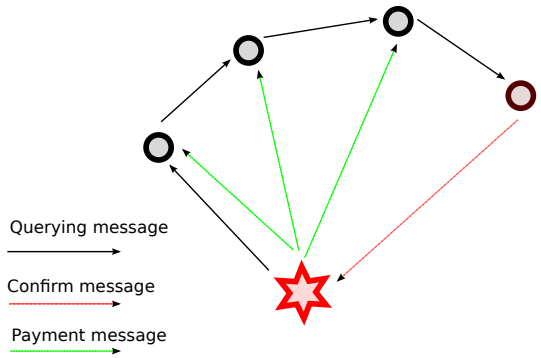

Figure 4: The chain/broadcast design

4 Peer Discovery

A peer needs to receive game state messages from different neighbors when it is in different regions, since the set of peers in its AOI keeps varying due to the movements of the peer and other neighbors nearby. It is important for players in the MMOGs to discover new neighbors when it joins the game or moves to a new region in the game world. (TODO: cite, to show that the peer discovery can impact the consistency and latency)

4.1 The protocols

When a peer joins the game, it needs to find the neighbors nearby in the game world.

When a peer moves to a new region, i.e., it already has some neighbors and try to make use of these neighbors to find new peers in the new region where it moves in. It sends a querying message to all the peers

When a peer first joins the game:

4.1.1 The broadcast neighbor discovery

On finding the neighbors, the peer will send a querying message to all the peers close to it. We represent the querying message by $(SrcPeerAddress, TTL, Position, Radius, Direct)$. On receiving a querying message, a peer will process as follows:

1. **Connect**: Check the *Position* and *Radius*, and connect to the requesting peer if (1) the distance with the requesting is less than *Radius*, and (2) they haven't been connected yet.

2. **Broadcast**: Check the TTL, *Position*, *Radius* and *Direct*, send the message when $TTL \geq 0$, to neighbors who are (1) in the region bounded by *Position* and *Radius*, (2) some other peers who are outside the region (*Position*, *Radius*) but in the extension of *Direct*.

4.1.2 How the contract and game points are transfered

(todo: remove) How the contract and game points are transfered

Value of a new neighbor: on receiving a new neighbor, the requesting peer gives it a value as discussed in Sec. 2.3. The price of a neighbor is decided by its importance (distance is the critical factor). Here we give the contract and game points the requesting peer should pay:

$$\begin{cases} \frac{a_2}{\log(dis(j,n))} & \text{The contract transfered between friends} \\ \frac{\alpha}{f_{in}} \cdot \frac{a_2}{\log(dis(j,n))} & \text{The game points among strangers} \end{cases}$$

Here, j is the ID of the new found neighbor, n is the requesting peer and i is a peer in the payment chain.

Payment chain: as mentioned above, a peer checks the querying message to decide whether need to send a connect message to the requesting peer. If it needs to connect to the requesting peer, it will send the requesting peer a "payment chain", which is a peer list $\{p_1, p_2, \ldots, p_M\}_M$. This chain contained in the message records the peers through which this message has come. When a peer re-send the message to others, it will first add its information to the chain in the querying message.

When the requesting peer finally receives a connect message from a neighbor, it will do the following work:

1. **Confirm**: the peer will send an acknowledgment message back, so that the peer will not re-try to connect any more.

2. **Pay**: the peer will find the payment chain in the connecting message, and should pay the peers in the chain list. The payment mechanism is similar with that for relaying contribution. The difference is that the requesting peer will send to all the peers in the chain, who may either be friends or strangers.

Table 2: Format of peer discovery message

Requesting peer ID	TTL	Position	Radius	Moving direction	Payment chain

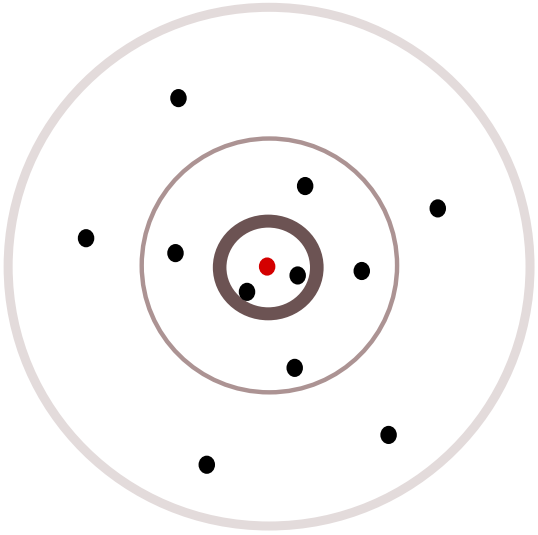

Figure 5: Neighbors in the first circle are the closest neighbors, who may request the peer's game state messages. Neighbors in the second circle are the full maintain neighbors, they don't exchange game states, but the neighbor list is complete. The third circle is the far neighbors, which are randomly maintained for neighbor discovery, the larger the distance is, the fewer neighbors are stored.

Format of querying message: the message contains TTL, peer position, moving direction.

todo: probability broadcast

4.2 The peers to return to the requesting peer

What neighboring information a peer stores:

4.3 Backup Schemes for The Missing Neighbors

(todo: remove) this subsection discusses the backup scheme

5 Evaluation

Consider free rider?

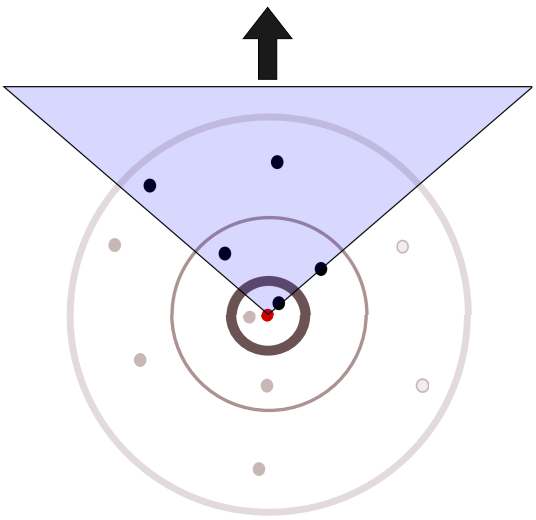

Figure 6: Neighbors in the circles are stored with directions. When a peer moves into a direction, it will ask the neighbors in that direction for the neighbor information, in return, the neighbors also calculate the best fit peers to return to the requesting peer. The nodes in black will be requested for the neighbors held in their neighbor information.

6 Conclusion

7 Appendix: problems

todo: probelms:

1. no global optimization?
2. quantify friendship
3. the problem of service peer

References

[1] D. Pittman and C. GauthierDickey. A measurement study of virtual populations in massively multiplayer online games. In *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, pages 25–30. ACM, 2007.