Crossroads
The ACM Student Magazine

Virtual Communities and Team Formation

by ***Yanru Zhang*** and ***Michael Weiss***

Introduction

With the growth of global computer networks, virtual communities have become an important new way for people to interact. People are beginning to realize that networks are not only affecting the way businesses operate, but also our everyday lives [7]. One of the simplest examples of a virtual community is online chat. Through a chat application, one can participate in diverse discussions with numerous people, many of whom are strangers.

A **community** has been defined as "a social grouping that exhibits to varying degrees: shared spatial relations, social conventions, a sense of membership and boundaries, and an ongoing rhythm of social interaction" [14]. Based on the meaning of the word "virtual" (i.e., computer-mediated), a **virtual community** can be understood as a social grouping that emerges from the webs of personal relationships embedded in computer networks. It describes the union between individuals or organizations who share a set of common interests and use electronic media to communicate with each other. Their ability to communicate is not restricted by time or location [15].

The concept of a virtual community has been adopted in many different application domains such as e-commerce (online auctions), e-learning (virtual classrooms), and e-gaming (multi-player games). Consider the e-commerce application of purchasing a product on eBay. The buyers and sellers interested in that product temporarily form a community that advertises the product, negotiates its price, and coordinates the transfer of ownership. Alternately, in e-learning environments, registered distance learners with an online university will meet both professor and classmates in a virtual classroom. In online gaming, one meets with friends online to play multi-player games, accumulate points, or improve basic skills.

Recently the concept of **collaborative virtual environments** has attracted considerable attention. Collaborative virtual environments have been defined as "computer-based, distributed, virtual spaces or sets of places" [4]. In such places, members of a community can meet and interact with each other, with software agents, or with virtual objects. Access to this kind of community is not limited to desktop devices, but might well include mobile or wearable devices, public kiosks, etc. [4]. In a collaborative virtual environment, community members will pursue both their own as well as social goals. With common social goals, members will adjust their individual goals, so as to contribute toward making the community, as a whole, more attractive. In order to achieve their common goals, the members need to collaborate and coordinate their activities.

In the following sections, we first describe the basic characteristics of virtual communities. After that, we discuss the concept of coordination and present our joining and leaving protocol for virtual game communities. We subsequently propose a coordination model based on layered reactive tuple spaces. Finally, we apply this coordination model to the domain of online multi-player games. We chose multi-player games as our application domain because games promise to become one of the most important applications of collaborative virtual environments.

Basic Characteristics of Virtual Communities

Virtual communities are intended to aggregate people, and to provide them with an engaging environment in which they can interact with others. People can find out who the other members of a community are, which community members have some special expertise, and what other people in the community are doing. It allows them to coordinate their activities with those of others. Communities encourage people to share knowledge and increase their mutual preferences. In the online auction example, for instance, sellers have more opportunities to expand their markets, while buyers can extract more value from sellers. What then are the basic characteristics of a virtual community?

Awareness

Awareness is defined as the ability to maintain and constantly update a sense of our social and physical context. Awareness is an essential precondition for making contact with other members of a community, since we need the information of who is in the same virtual place, who shares common interests with us, and who is available for collaboration when we join a community [14]. Increased awareness encourages informal spontaneous communication between community members, and contributes to their ability to make informed decisions.

Privacy

Awareness needs to be balanced with privacy. If we consider common interests as the basis for the formation of a community, it is sometimes difficult to find users who share our interests on the network. The reason for this is that people do not like to post too much of their personal information in a publicly accessible place, or do not update their profiles as their interests change. We need to address this issue by not only providing a means for safely and securely introducing users with similar interests to each other, but also by protecting their privacy.

Trust

The stability of a community depends on the right balance of trust and distrust [1]. We are faced with information overload, increased uncertainty, and risk-taking as prominent features of modern life. As members of a society, we cope with these complexities and uncertainties by relying on trust. Trust is the basis for all social interaction. As a recent survey on trust in e-commerce summarizes, "Trust is about social relationships, and about building networks that deliver what they promise, be it a product, a collaboration, or simply reliable information" [10].

Knowledge Sharing

Differences in knowledge encourage people to communicate. Knowledge-sharing facilitates the formation of a virtual community. It also plays an essential role in the maintenance of the community so that people can benefit from participating in the community. This will, in turn, attrract more people to join, and thus contribute to the growth of the virtual community. The richer the knowledge accumulated in a community, the more interesting the community becomes for its members, and the more difficult it will be for members to leave.

Collaboration and Coordination

Collaboration and coordination are required when forming a community, pursuing individual goals while meeting community goals, and changing the structure of a community. Community members have incomplete knowledge of their partners and their environment. This forces them to collaborate and coordinate with each other. In distributed Internet applications, participants can furthermore choose to collaborate and coordinate directly or indirectly. Each mode has different benefits according to the application domain (which elaborated on later in this article).

Virtual Communities for Online Multi-player Games

Online multi-player games are considered one of the promising applications of collaborative virtual environments. A game can be defined as a simulation that "works wholly or partly on the basis of a player's decisions" [11]. Online games can be grouped into two main categories [8, 11]: classical games such as board games (chess, checkers, etc.), and pure computer games such as action, role-playing, adventure, strategy, god, and team sport games. Players meet online and form **virtual game communities** for different kinds of games based on their interests, skill, and reputation.

Coordination in Virtual Game Communities

Everyone has an intuitive understanding of coordination, and can give real life examples. Coordination is what turns the players in the hockey game into a team, or individual instrumentalists into an orchestra. One definition is that coordination is about "managing dependencies between activities" [9]. This definition focuses on the interdependence between activities (the coordinated moves of hockey players, or the synchronized performance of complex scores). Other definitions consider coordination the very foundation of programming. Gelernter [6] goes as far as saying that "Programming = Computation + Coordination". This means that separating computation from coordination allows not only the reuse of computational components, but also that of coordination logic. We can conclude that coordination involves active entities that interact with each other at certain times and in certain places following agreed-upon rules.

How can we apply the concept of coordination to virtual game communities? If the game players pursue their own goals, based on their individual interests and skills, why do they still need to coordinate their activities with other players? The answer lies in considering how gaming has evolved from single-player games to multi-player games with cooperative goals. There are three motivations for using coordination in multi-player games:

- In a single-player game, successful completion mainly depends on the player's skill. However, with the "popularity of network play growing every day, modern game play is moving away from single player mode to team based game with cooperative goals" [13]. Game developers are using teams as fundamental parts of their design. In an online game, each player has incomplete information about the state of the game, and the other players in the game. Players need to help each other to complete the game, exchanging their partial views of the game, suggesting moves to each other, and reassigning the workload among themselves.
- Coordination can make games more appealing. In the early development of games, game developers focused mainly on visual effects. However, today players expect more than exciting graphics and surprising plots. They are paying more attention to being able to find other players with similar personalities and interests, to changing a game dynamically to meet their own needs (e.g., SimCity), and to how they are perceived by other players (i.e., their reputation as a team player).
- If coordination is not explicitly represented in the game, this does not mean that coordination is not needed. For example, the actions of non-player characters need to be coordinated with the moves of human players. Making coordination explicit, and separating it from computational components can help in avoiding duplication of effort in developing non-player characters from scratch each time. Similarly, the rules of a game, if encoded as coordination rules, can be adapted more easily to create new games from existing ones.

Protocol for Joining and Leaving a Community

The term protocol, in its everyday sense, refers to a system of rules governing formal occasions. In technical terms, a **protocol** is a set of conventions governing the handling and especially the formatting of data in an electronic communications system. For example, in a virtual community, everyone has the right to join the community, make contributions to the evolution of the community, and leave the community at any time. However, virtual game communities, like physical communities, still need social rules to govern the behavior of players. For instance, while a player is free to leave a game prematurely, this may be frowned upon by the community.

In a virtual game community there is, conceptually, one large community consisting of all game players, which is, in turn, composed of many smaller communities or subcommunities for different kinds of games and different skill levels of players. Players differ in their experience, domain knowledge, skill level, interest, and reputation. For example, some players are beginners, while others play at an advanced level; some prefer action games, while others want to play only strategy games; some have received high ratings from their peers, other may not have been rated at all.

In our protocol, each virtual game community has a community leader, represented by a **leader agent**, whose task is to control access to the community, maintain the reputations of its members, and to interact with the leaders of peer communities at the same level. Subcommunities for the same kind of game each have their own leader. Every community leader can communicate with other leaders to share knowledge, make recommendations on which new members can join, or might be suitable to join another team and coordinate the re-formation of the team, should a member leave.

Upon joining a virtual game community, each player will be represented by a **player agent**. They first enter the lobby where they can check how many kinds of games can currently be played and which players have joined which games (**Figure 1**). Newcomers could also learn about the different types of games (e.g., the game rules), and obtain information on joining or leaving a game from the lobby. Based on the information they received, their description of the other team members (skill level, reputation, etc.), and their own skill level, players will select a subcommunity to join.

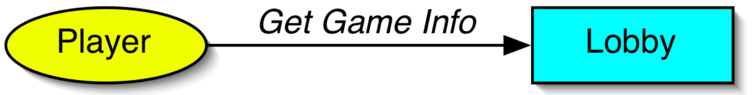


Figure 1: Player connecting to the lobby room.

Within the subcommunity for a single game, players know the identities of the other players. They know with whom they can and should share knowledge. After the team has been formed, everyone will make contributions to common team goals. Players will be rated based on their effort, contribution to the game, and the final result of playing the game. The more contributions a player made, the more credits she will receive towards her reputation level.

One characteristic of teams in an online game community is their dynamic nature. Cooperation and coordination does not only exist among the players in a single community, but also between communities. Suppose that in a battle game one team needs to add more soldiers or soldiers with specific skills while the game is in already progress. The leader of the team could ask other players under the leader's control or other leaders of different communities for assistance.

Leaving a virtual game community can be classified either as normal leaving, or urgent leaving. Normal leaving, after the game has terminated, will not affect the progress of the game. When a player plans to leave after the game is over, she will send a leave request to the community. The leader agent will get the player's information, announce her departure to the other members remaining of the team, and record her rating.

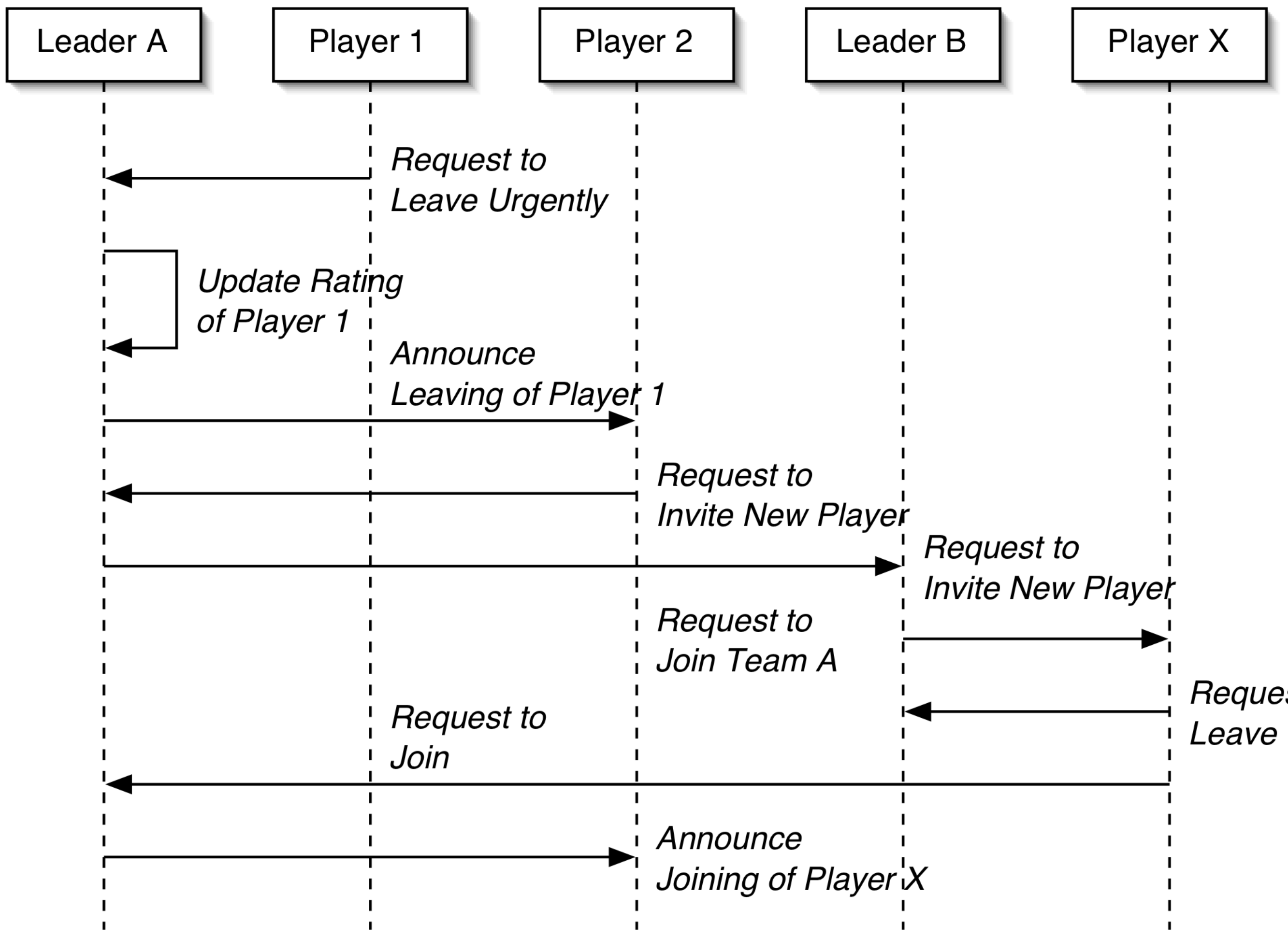

Leader A
Player 1
Player 2
Leader B
Player X
Request to Leave Urgently
Update Rating of Player 1
Announce Leaving of Player 1
Request to Invite New Player
Request to Invite New Player
Request to Join Team A
Request to Leave
Request to Join
Announce Joining of Player X

Figure 2: Protocol for leaving the game community.

In the case of **urgent leaving**, a player wants to leave the game before the game is finished (**Figure 2**). The leader agent will ask if this player can stay and continue the game, accompanied by some kind warning of the consequences of urgent leaving. This player's insistence to leave will result in a decrease of her rating for future attempts to join the community. New candidates to replace the leaving player can be found in one of several ways:

- At the time a player leaves, a new player joins the team and replaces the former one to continue the game as if nothing had happened. This player will continue the state left by the leaving player.
- The remaining players could send an invitation to players who are kept on a waiting list by the leader agent. The invited players are selected based on their experience, skill level, interest, ratings, and other factors important to the invitees. If the selected player agrees to join, she joins the team as usual.
- The leader agent coordinates with players in other teams or other leader agents for the same kind of game. If the right candidate has been found, she will be asked to join the team and given a new job according to the dynamic team formation. Also, through coordination with other team leaders, community leaders could recommend where to find the right player (they are aware of who knows who and who knows what).
- If no right candidate could be found, or upon the request of other players, the leader agent will create a software agent to replace the leaving player. Later, the software agent can be replaced by a human player.

In this way, the game will not be disrupted or maliciously destroyed by the decision of a player to leave the game prematurely. The reputation system helps constrain player behavior to follow both game rules and social rules. It is also an incentive for players to contribute to a game community. Suppose John is a beginner for the game Bridge, but he selects to join an team of advanced players instead. As a result, he could not complete the game adequately, or collaborate with others harmoniously. His final score will be affected, as well as his rating.

Coordination Model for Virtual Game Communities

There are three important components in a coordination model:

- **Coordination entities:** These are "actively computing entities" [2], often programmed using agents. Agents have their individual goals, as well as social goals. For individual goals, agents can choose their own way to achieve them. However they need to use a coordination mechanism or coordination channel to communicate or negotiate with other agents on how to accomplish social goals, if they share the same activity and common goals. In a virtual game community, coordination entities include player agents representing players, and software agents replacing a leaving player or leader agent. In various kinds of games, agents exchange their actions, intentions, knowledge, locations, and decisions to coordinate with each other for reaching the community goal.
- **Coordination media:** Entities coordinate themselves either by direct or indirect coordination. Coordinating directly means that entities may transmit message over some forms of channels [2]. For this to work, they need to know the other identities of other agents in advance, where others they are located, and when they can be reached. Peer to peer communication such as the message-passing paradigm underlying sockets is one of the examples using the direct method to realize coordination. In contrast, indirect coordination media uses the idea of a shared data space. Its main design goal is to provide a persistent data repository that uses an associative addressing mechanism to realize anonymous communication among the coordination entities. Agents communicate by sending data to the shared data space while interested partners can receive data by registering with the space. There could be a single data space, multiple data spaces, or nested data spaces depending on the requirements of the application. The shared data space model is more suitable for today's Internet applications due to its open, dynamic, mobility, and heterogeneous environment.
- **Coordination rules:** Coordination rules are the laws used to govern the relationship between entities and coordination media. One way to embody coordination rules is to embed them inside the agents, but there are some disadvantages to that. Agent designers need to anticipate all future situations that an agent may meet, in order to define the proper action or response to changes in the environment. For instance, in a strategy game, the agent could lose the game if one of the strategies of this agent is missing from its design. From a reusability and scalability point of view, it is not a good choice to embed social rules inside the agents. Each time the game rules are changed, or the communication protocol is modified, no matter how small the change, each agent involved in the game would need to be rewritten to reflect the change. Another way is to use programmable tuple spaces, which separates the concerns of coordination and computation. We will apply this to our coordination model, so that the developers of player agents don't need to be concerned about coordination rules. Players in online game community have common interests and

similar skill levels or proficiencies. They are eager to share their experiences and emotions, and share their knowledge about the game. They try find partners with similar interests, skill level, and good reputation. All players should abide by the rules of the game and social rules. They coordinate their activities in order to reach common goals: winning the game and receiving high scores and good ratings.

The Linda model uses a shared data space as a coordination medium [5, 6, 12]. It allows producers and consumers of information to put and retrieve data via a **tuple space**. Data is put into a tuple space in the form of tuples. Tuples are essentially ordered collections of primitive data types. The Linda model is suitable for use in open distributed systems due to a variety of attractive features [17]:

- **Identification uncoupling:** The producer of a tuple does not need to know who consumes it, and the consumer does not need to know who produces it. Tuple space communication is anonymous.
- **Spatial uncoupling:** Tuple spaces use an associative addressing scheme based on pattern-matching. Tuple spaces provide a globally shared data space to all processes or agents regardless of machine or platform boundaries, as long as they have access to the space.
- **Temporal uncoupling:** Tuples have their own life span, independent of the processes that produced them, or any processes that may consume them. This enables temporally disjoint processes to communicate in a seamless manner.

In our game community, there is at least one **local tuple space** for the player agents of each game team. All player agents who have access to this space will coordinate their activities via this shared repository. In practice, a single tuple space is not enough. We need **multiple tuple spaces**, which means new spaces can be created as needed, depending mainly on the number of players, and the number of games. For example, in a Yahoo-style card game [18], each room could use a separate local tuple space to coordinate between two and four players.

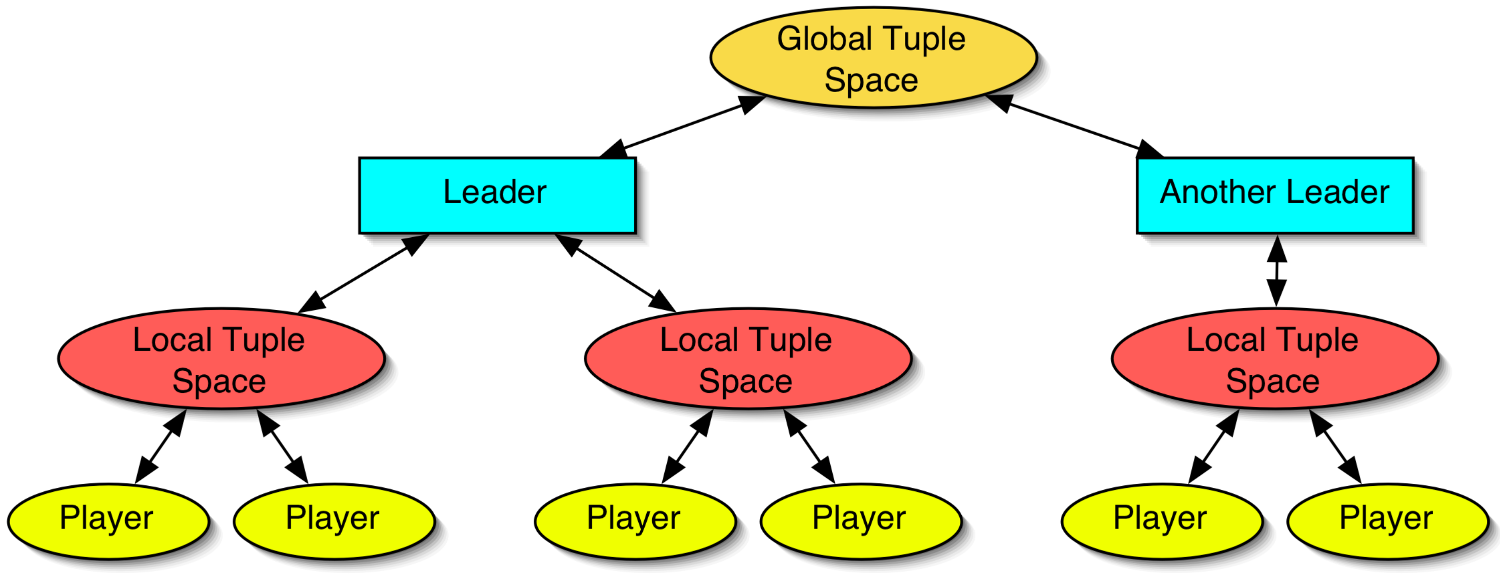


Figure 3: Layered Reactive Tuple Spaces.

The original Linda tuple space had no "control" over the operations on the space. It is simply a persistent repository for shared data. In our model, we add some "life" to the tuple space by making it reactive. In a similar manner to [3], we create a **reactive tuple space** by allowing reactions to be associated with specific operations on the space. A reaction is a piece of code that is fired when the associated operation is invoked. Similar to tuples, reactions reside in the tuple space. We use these reactions to embed social rules in a space. An example of a reaction is to automatically broadcast a `join` tuple to all other players whenever a `join` tuple is written to the space by a new player.

In order to support coordination between community leaders, we propose an extension to the basic reactive tuple space model called **Layered Reactive Tuple Spaces** (LRTS). This extension introduces multiple layers of tuple spaces. For example, in addition for the tuple space for individual games, there can be a global data space for each kind of game (**Figure 3**). This global tuple space is used to allow leader agents to communicate on issues of team formation, leaving of a team member, and rating players. Leader agents have access to both local spaces and the global tuple space, while player agents can only access the local tuple spaces for their respective games.

Case Study: AgentWorld

Many online multi-player games are competitive. Cooperation is still possible in those games, and often of decisive strategic importance. For example, consider a battle game, in which temporary alliances against a mutual opponent can benefit all members of the alliance. However, richer cooperation behavior can be studied more conveniently in the context of a cooperative game. As cooperative games require that all players work towards a common goal, the notion of a social goal is already embedded into the game by design.

Here we choose a cooperative game called AgentWorld [16] as a case study for applying our coordination model to see how team members coordinate their actions at the beginning and end of the game. A game board of AgentWorld is a grid filled with pieces (squares) of different color, destinations (circles), and representations of the players, as shown in **Figure 4**. In the basic scenario, the whole game board is visible to each player. An advanced version of the game can also be played in which only a portion of the game board is visible to a player. Thus we can model the situation that players only have partial information about the game environment.

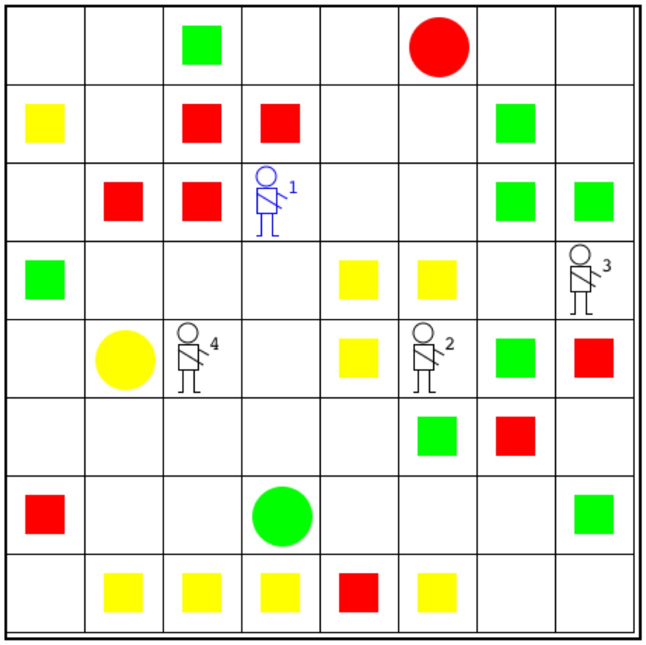


Figure 4: Each agent represents a player who participates in the game AgentWorld.

The objective of the game is to move all pieces to their correct destination (which is the destination of the same color as the piece) in a minimal number of moves. Only one player (the currently active agent is shown in blue) at a time can make a move to a field next to him. Making a move means that a player can step to a free field, pick up a piece, or put down a piece. Because each move will increase the total number of moves by one, all players try to keep the total number of moves low. The common goal for the players then becomes to coordinate their moves with those of others to minimize the total number of moves expended. There are various possible strategies players can follow, for example, a simple "relay" strategy in which agents move pieces along a chain (passing them from one to another), instead of each agent executing the required moves individually.

When we apply our coordination model to this game, we will use the following types of tuples:

- `player` tuple: contains a player's unique id used to refer to a player
- `join` tuple: posted by a player when joining a game
- `begin` tuple: notification to all waiting players that a game has the required number of players
- `game` tuple: contains game status information
- `move` tuple: indicates a move on the game board
- `message` tuple: contains a message sent among the players

Selecting and Joining a Team

When the lobby server starts, it will start a certain number of game servers (**Figure 5**). Each game server will host a copy of the AgentWorld game, but with a different configuration of game parameters such as the number of players, the number of pieces to move, the view size, as well as how many players are currently waiting, and their identities.

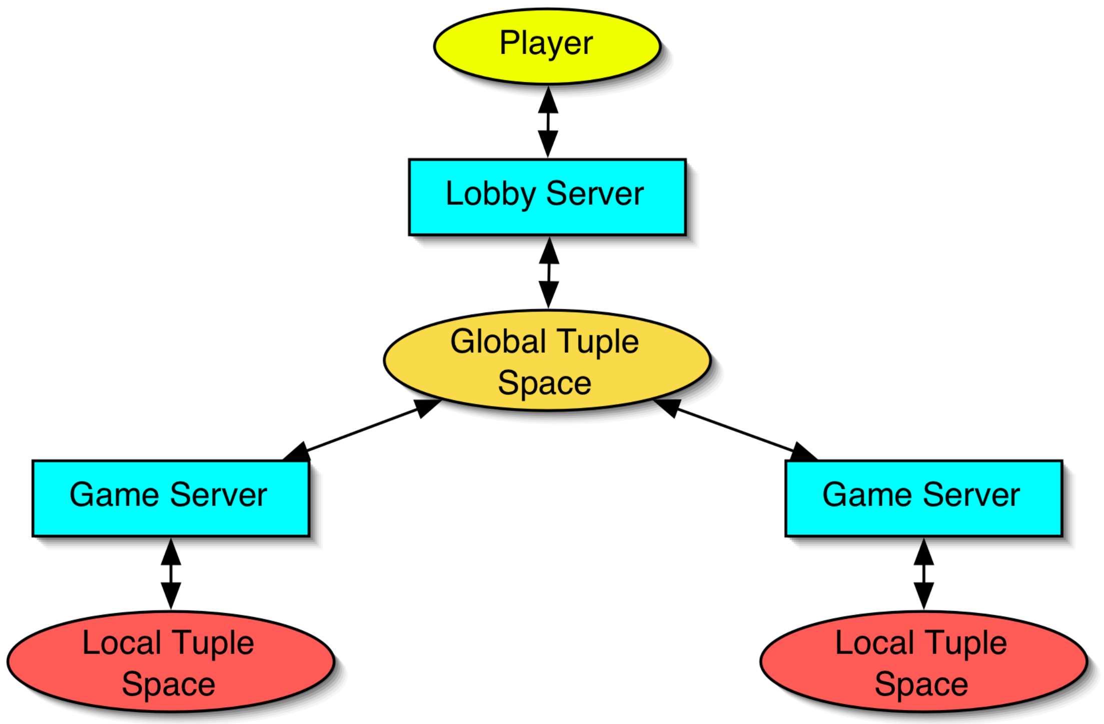


Figure 5: Players get game status information through the global tuple space.

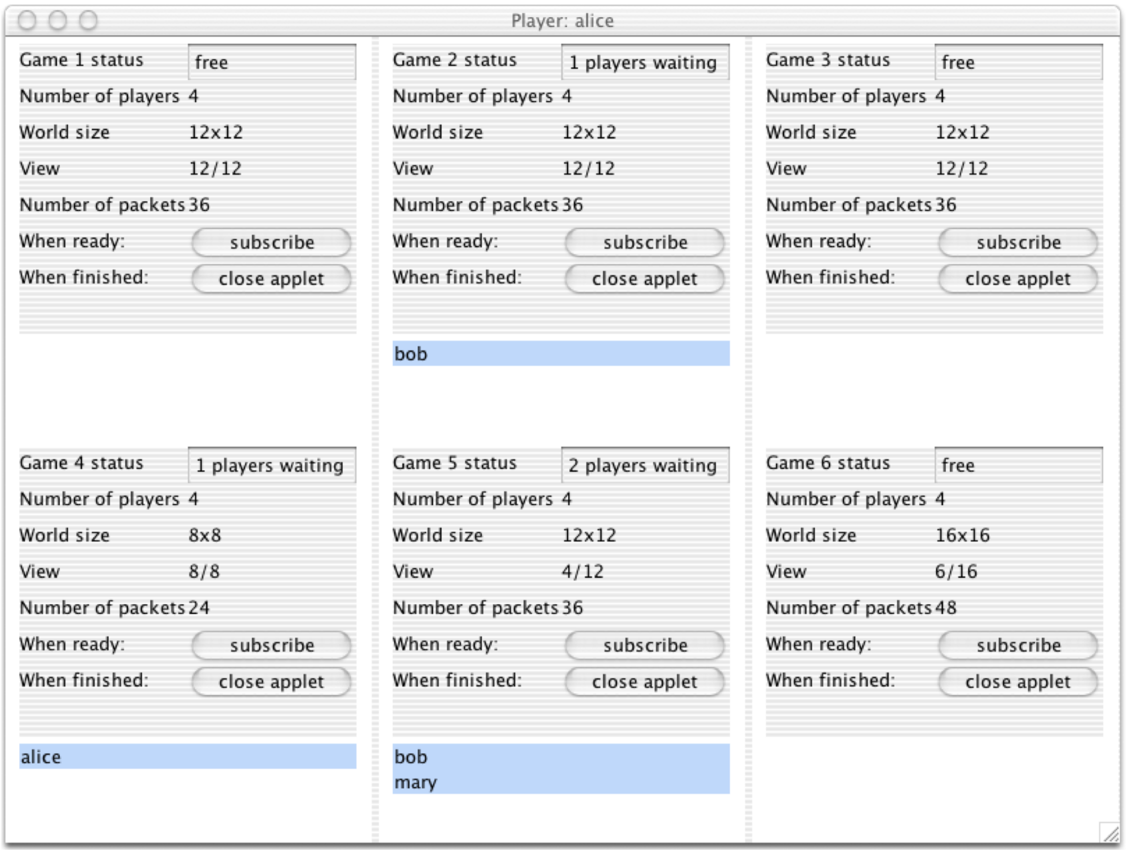
On start-up, each game server will write a game tuple to the global tuple space:

```
Tuple tuple = new Tuple ("game");
tuple.add (new Field ("name", name));
// configure other game information
globalTupleSpace.write (tuple);
```

A player who wants to join the game must connect to the lobby server. As soon as the lobby server receives a request from the player to connect, it will read all current game tuples from the global tuple space:

```
Tuple template = new Tuple ("game");
template.add (new Field ("name", String.class));
// add more info it needs
return globalTupleSpace.readAll(template);
```

The player will then be presented with a lobby interface showing the status of each game in progress (**Figure 6**). From the information he received, the player will decide which game to enter. This decision will also depend on information such as the player's skill level, the reputations (ratings) and skill levels of other players who have already joined a particular game, etc. This information is not shown in the lobby interface in **Figure 6**, but we are working on ways of visualizing such "rich" information in a compact manner.


Player: alice

Game 1 status	free
Number of players	4
World size	12x12
View	12/12
Number of packets	36
When ready:	subscribe
When finished:	close applet

Game 2 status	1 players waiting
Number of players	4
World size	12x12
View	12/12
Number of packets	36
When ready:	subscribe
When finished:	close applet

bob

Game 3 status	free
Number of players	4
World size	12x12
View	12/12
Number of packets	36
When ready:	subscribe
When finished:	close applet

Game 4 status	1 players waiting
Number of players	4
World size	8x8
View	8/8
Number of packets	24
When ready:	subscribe
When finished:	close applet

alice

Game 5 status	2 players waiting
Number of players	4
World size	12x12
View	4/12
Number of packets	36
When ready:	subscribe
When finished:	close applet

bob
mary

Game 6 status	free
Number of players	4
World size	16x16
View	6/16
Number of packets	48
When ready:	subscribe
When finished:	close applet

Figure 6: The lobby interface shows the status of each game.

Beginning a Game

Once a player joins a game, its player agent has the access to the local tuple space for that game. All actions the player performs will be coordinated with other players through this local tuple space such as all of his moves on the game board, and his messages exchanged with other players. Each request from a player to join a game will cause a `join` tuple to be written to the local tuple space (**Figure 7**). The local tuple space will then check if the required number of players have joined the game, and if so post a `begin` tuple to the local tuple space that notifies all players that the game can start.

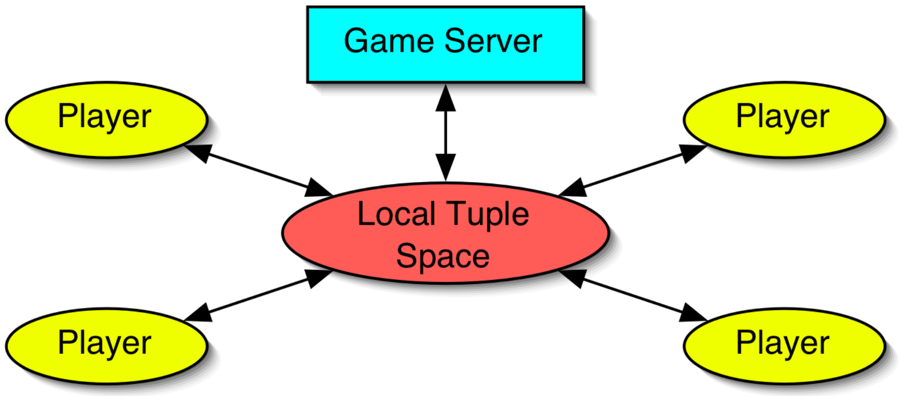


Figure 7: Players coordinate with each other via a local tuple space.

On the player side, a `join` tuple will be posted to the local tuple space. The player will then block until a `begin` tuple has been posted to the tuple space:

```
Tuple template = new Tuple ("join", new Field ("name", name));
localTupleSpace.write (template);
//...
Tuple begin = new Tuple ("begin");
localTupleSpace.read (begin);
```

On the game server side, a reaction will execute to check how many players have already joined the game (including the player who has just joined), and post a begin tuple, if appropriate:

```
state = localTupleSpace.readNumber (new Tuple ("join"));
if (state == numberOfPlayers) {
  localTupleSpace.out (new Tuple ("begin"));
}
```

Inviting Other Players

As the sequence diagram in **Figure 2** shows, after one player leaves a community, the remaining players can invite players waiting in other game servers to re-form the team. This process involves the use of layered tuple spaces. As illustrated by **Figure 8**, Player 2 will first send the inviting request to the local tuple space, where it is read by the leader agent for the game (Leader A). This leader agent will write the invitation tuple to the global tuple space, from which it can be read by other leader agents (e.g., Leader B). Leader B may forward the invitation to a player under its control such as Player X. The details of the inivitation protocol are beyond the scope of this paper.

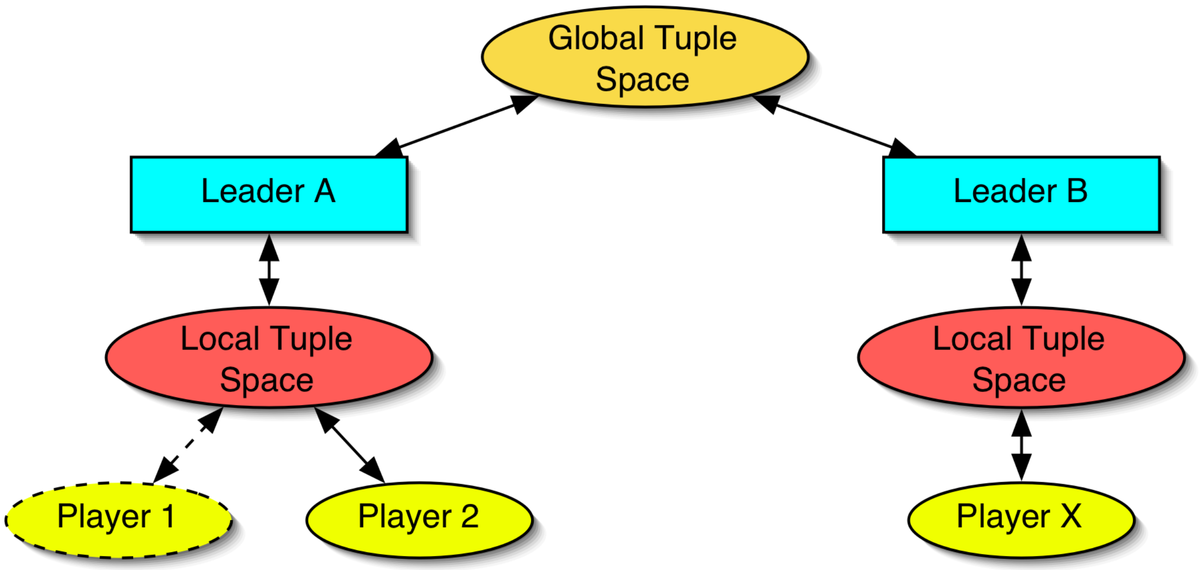


Figure 8: Inviting a player via a layered tuple space.

Conclusion

Virtual communities for gaming have become a focus of significant research effort. Coordination plays an important role in the formation, evolution, and dissolution of a game community. Adding coordination to online multi-player games makes them more appealing. Players can choose their favorite games based on their interests, skill level, and other players they know. We analyzed the needs of coordination in the e-gaming domain, and defined a coordination model based on Layered Reactive Tuple Spaces (LRTS). We also presented results of an initial implementation of the model.

However, we believe that our coordination model is not limited to applications in multi-player games. Coordination mechanisms play a central role in any type of collaborative virtual environment. One application domain of great interest for our future work is e-learning. Similar problems arise from managing groups of users, matching up online classes with learners, and supervising the progress of a class. A direct application of the results from our study of the e-gaming domain is even possible, since games are a popular way of imparting learning material.

References

1
Abdul-Rahman, A., and Hailes, S. *Supporting Trust in Virtual Community.* Hawaii International Conference on System Sciences (HICSS), IEEE, 2000.

2
Busi, N., Giancarini, P., et al. *Coordination Models: A Guided Tour.* in Omicini, A., Zambonelli, F., et al (eds.) Coordination of Internet Agents: Models, Technologies, and Applications, 6-24, Springer, 2001.

3
Cabri, G., Leonardi, L., and Zambonelli, F. *Reactive Tuple Spaces for Mobile Agent Coordination.* Workshop on Mobile Agents (MA), LNCS 1477, Springer, 1998.

4
Churchill, E. (ed.) *Collaborative Virtual Environments*. Springer, 2001.

5
Freeman, E., Hupfer, S., and Arnold, K. *JavaSpaces: Principles, Patterns, and Practice.* Addison-Wesley, 1999.

6
Gelernter, D., and Carriero, N. *Coordination Languages and their Significance,* Communication of the ACM, 35(2), 97-107, February 1992.

7
Ishida, T. *Towards Computation over Communities,* Community Computing and Support Systems, LNCS 1519, 1-9, Springer, 1998.

8
Laird, J. E., and van Lent, M. *Human-level AI's Killer Application: Interactive Computer Games,* Proceedings of the National Conference on Artificial Intelligence American Association of Artificial Intelligence (AAAI), 1171-1178, 2000.

9
Malone, T., and Crowston, K. *The Interdisciplinary Study of Coordination,* ACM Computing Surveys, 26(1), 87-119, March 1994.

10
Merrill Lynch. *E-Commerce and the Challenge of Trust.* April 2003. <**http://www.ml.com/woml/forum/ecommerce1.htm**>, April 2003.

11
Niederberger, C., and Gross, M. *Towards a Game Agent,* Technical Report 377, Institute of Visual Computing, ETH Zürich, 2002.

12
Omicini, A., and Zambonelli, F. *Tuple Centers for the Coordination of Internet Agents,* ACM

Symposium on Applied Computing, ACM, 1999.

13
Rabin, S. (ed.), AI Game Programming Wisdom. Charles River Media, 2002.

14
Schlichter, J., Koch, M., and Xu, C. *Awareness: The Common Link Between Groupware and Community Support Systems,* Community Computing and Support Systems, LNCS 1519, 77-93, Springer, 1998.

15
Schubert, P. *The Pivotal Role of Community Building in Electronic Commerce.* Hawaii International Conference on System Science (HICSS), IEEE, 2000.

16
Weyns, D. *Socialilty in Multi-Agent Systems.* April 2003. <**http://www.cs.kuleuven.ac.be/~danny/SocialityInMAS.html**>.

17
Wyckoff, P., McLaughry, S., et al. *T Spaces.* IBM Systems Journal, 37(3), 454-474, August 1998.

18
Yahoo Games, <**http://games.yahoo.com**>.

Biographies

Yanru Zhang (**yanru_zhang@yahoo.com**) is a Master's student at the School of Computer Science, Carleton University, Canada. She holds a BEng from Beijing Union University in 1995. Her main areas of interest are distributed systems, Internet applications, coordination in virtual communities, and agent-based systems.

Michael Weiss (**weiss@scs.carleton.ca**) is an assistant professor for electronic commerce at the School of Computer Science, Carleton University, Canada. He received his PhD degree from the University of Mannheim, Germany, in 1993. His research areas are intelligent agents, electronic commerce (in particular, web services, and virtual communities), and software patterns. Prior to joining academia he was most recently manager and technical lead at Mitel for a commercial agent-based infrastructure for service provisioning.