Independent Submission
Request for Comments
Category: Experimental

Shan Han
The George Washington University

October 2018

Online Card Game “Love Letter”

Abstract

The document describes a system and its system which users can play card game “Love Letter” online. Users play game according to rules and win scores, user with the highest score at the end of game wins the game. The protocol depends on TRANSMISSION CONTROL PROTOCOL(TCP) and INTERNET PROTOCOL(IP) for transmitting data between hosts.

Status of This Memo

This document is published for experimental purpose, it specifies online card game “Love Letter”. It is for examination, experimental implementation, and evaluation.

This document is product of project of CSCI6431 Computer Network of The George Washington University. This RFC represents individual opinion in the project.

Table of Contents

1. Introduction

1.1 Motivation

Card game is prevalent in the world. Players play with friends for entertainment and communication. A significant number of games are designed every year, “Love Letter” is one of them. The game was introduced in May 2012 by Seiji Kannai. Players defeat each other by skills of roles in game. Game includes many rounds, last standing player is the winner in the round and win one score. Player who wins target score first wins the game.

However, due to players may have time conflict and they may live far from each other, it is hard to call players together. Hence, network is the best solution for the problem. People can play the game online.

1.2 Game Rule

1.2.1 Background

The game allows 2 - 4 players. Every player is attracted by beauty of princess Annette and they try to send love letter to the princess. Unfortunately, princess Annette locks herself in the palace. Players have to deliver love letter with assistance of collaborators.

1.2.2 Collaborator

There are 8 kinds of collaborator. Every kind of collaborator has unique skill and strength points. In addition to, amount of each kind of collaborator is different. Players do not know collaborator of each other.

Guard
Guard has 1 strength point. The number of guard is 5. Player with assistance of guard can choose one player to guess identity of collaborator. However, guard can not be a option of guess. If guess of the player is correct, target player would be defeated in current round, or game would continuos.

Priest
Priest has 2 strength points. The number of priest is 2. Player with assistance of priest can choose one player to know identity of collaborator.

Baron
Baron has 3 strength points. The number of baron is 2. Player with assistance of Baron can choose one player to compare strength points of collaborators privately. Player with lower strength points is defeated in current round.

Handmaiden
Handmaiden has 4 strength points. The number of handmaiden is 2. Player with assistance of handmaiden can not be effected by skills until next turn.

Prince
Prince has 5 strength points. The number of prince is 2. Player with assistance of prince can choose one player to discard his/her collaborator and target player has to find a new collaborator.

King
Kind has 6 strength points. There is 1 king. Player with assistance of king can choose two players to exchange their collaborators.

Countess
Countess has 7 strength points. There is 1 countess. Countess has no skill, if player has collaborators countess and king or princess, countess must be abandoned.

Princess
Princess has 8 strength points. There is 1 Princess. If player abandoned princess for any reason, the player loses current round.

1.2.3 Game Rules

There are 16 cards. Every card represents a collaborator. One random card is discarded from 16 cards each round and it is unknown. The game includes many rounds. At beginning of round, every player has one card and rest of cards which are unknown to players are placed for drawing. Then player draws and uses card one by one. During each turn of player, player draws one card from rest of cards, then the player has 2 cards. Next the player choose one card to use skill and the card is discarded. Player who is defeated by skill or discards princess card loses current around. There is only one winner each round. If there is no card available for drawing, player with the highest strength points wins current round. Winner gets one score. Player who wins target score first wins the game.

2. Structure

The game is design on client-server model. TCP and IP are used to deliver data between client and server. Server can not establish connection with more than 4 clients.

2.1 Client Server Model

2.1.1 Terminology

Username
User creates unique username when client connects to server. Users can distinguish each other by usernames. Username can be any format but it must not contain ‘-’, ’+’, ‘_’ and ’:’.

User code
Client has a unique user code, it is integer. Server and client can find corresponding client by user code. The range of user code is from 0 to 3.

Player status

Client has four player statuses. Every status has corresponding code. Firstly, ‘regular’ status means that client can be effected by skills, its code is 1. Second status is ‘ready’, client with ‘ready’ status is ready for game, its code is 2. Last but not the least, client with ‘defeated’ status loses current round. 0 is the code of ‘defeated’ status. Finally, player with ‘immune’ status can not be effected by skills. Its code is -1.

Target score
Target score is threshold of judging winner of game, client who reaches target score first wins game. Range of target score is from 3 to 7. Client decides target score.

Player score
Client would get one score if it wins a round. Range of player score is 0 to n. n is target score. When client gets n score, the client is final winner of game and game is over.

Card
Collaborator is represented by card. There are 16 cards and 8 kinds of card.

Card code
Every card has corresponding card code. Server and client can distinguish cards by card code.

Card code contains two parts. First part is called collaborator number, it represents collaborator. There are 8 collaborators, thus range of the collaborator number is from 1 to 8. The relationship between collaborator number and collaborator is shown by Table 1.

Collaborator Number	Corresponding Collaborator
1	Guard
2	Priest
3	Baron
4	Handmaiden
5	Prince
6	King
7	Countess
8	Princess

Table 1

Because each kind of collaborator has different amount, second part of card code is called identifier, it discriminates cards which stand for same collaborator. The relationship between identifier and card is shown by Table 2.

Collaborator	Amount	Range of Identifier
Guard	5	[10,14]
Priest	2	[20,21]
Baron	2	[30,31]
Handmaiden	2	[40,41]
Prince	2	[50,51]
king	1	60
Countess	1	70
Princess	1	80

Table 2

Two parts are combined by ‘_’. The representation of card code is ‘collaborator number_identifier’. For example, ‘8_80’ represents Princess.

Card status
Card has two statuses and they have corresponding code. First status is ‘regular’ status. Client can hold cards with ‘regular’ status, its code is 1. Secondly, ‘discarded’ status means that card is owned in current round. 0 is code of ‘discarded’ status.

Card name
Card name is name of collaborator. There are 8 card names, they are ‘guard’, ‘priest’, ‘baron’, ‘handmaiden’, ‘prince’, ‘king’, ‘countess’ and ‘princess’.

Card description
Every card has card description which describes skill of the card. The correspondence table is specified by Table 3.

Card strength
Card strength is strength point of collaborator. Card strength equal to collaborator number of the card.

Card name	Card description
guard	You can choose any player to guess card, if you are right, the target player lose, else nothing happen.
priest	You can choose any player to see card.
baron	You can choose any player to compare cards' strength, player with lower strength loses.
handmaiden	You can not be effected until your next turn
prince	You can choose any player to discard his/her card and he/she has to draw a new card.
king	You can choose any two players to exchange their cards.
countess	You must discard this card when you have 'king' or 'princess'.
princess	You must protect this card, try to keep the card to end of game.

Table 3

2.1.2 Client

The port number of client is 52014. Client is user-end application with graphic user interface. It displays username, player status and player score of clients and target score. In addition to, card name, card strength, card description of cards which the client holds and results of using card are shown. Data of client and card are retrieved from server.

Firstly, client can connect to server by port 52013. when connection is established and, client requests user to create unique username. If username is invalid or exist, client would request username until username is valid and unique.

Secondly, client stores username, user code, player score and player status of all clients. However, only usernames, player scores and player statuses are displayed.

During game, client should store card information which client holds. Card name, card strength and card description are displayed. In addition card code and card status are hidden.

Next, client can set player status to ‘ready’ or ‘regular’. When player status is modified, client should send notification to server. If client receives message of client information, it has to update client information which it stores according to the message.

Thirdly, client can use skill of card by sending message about skill to server in turn of the client. Skill can only be used to

client with ‘regular’ player status. Server handles the message and responds with appropriate results, the response means that turn of the client is over, then client receives updated client information from server and updates client information which is stored at the client.

Fourthly, if server finds that winner has to be selected by comparing card strength of survival clients, server would inform client. Then client displays the message to user. If winner is decided, server would send winner information and updated client information to all clients. Client demonstrates winner information and updates client information according to messages.

Fifthly, simple game rule can be displayed for reminding users.

Sixthly, client can quit game before game or after game. Client quits game by sending message to server. Correspondingly, client should update client information when it receives message of client information from server.

Seventhly, client communicates with server only, there is no communications between clients. Client sends requests to server, server processes them, then server responds request to sender or all clients.

Finally, messages which are transported between client and server are formatted, thus client must have abilities to parse and generate formatted messages. The message format will be described at commands section.

2.1.3 Server

52013 is port of server. Server stores username, user code, player status, player score of all clients, cards which every client holds and target score. In addition card code, card status, card name, card description and card strength of all 16 cards.

When client connects to server and sends username successfully, server would assign user code to username, set player status of client to ‘regular’ and set player score to 0. Next, server broadcasts information of all clients to clients. However, if server has been connected by four clients or game has been started, new client would be notified and it can not connect to server.

Secondly, server should update player status of client if client sends message of modifying player status and should broadcasts the message to clients, thus clients can update player status which is stored at them. Furthermore, server should count the number of client whose player status is ‘ready’. When all clients are ready, server starts game.

Thirdly, At the beginning of a new round, server should shuffle 16 cards and discard one of them randomly.

Fourthly, during game, server should distribute a card whose card status is ‘regular’ to client when it is turn of the client. Server

should send card code, card name, card strength and card description to clients. Then server modifies card status of cards which are distributed to ‘discarded’. Besides, server stores the card information and its owner.

Fifthly, turn starts from winner of previous round, if it is first round, server would choose client randomly and would start turn from the client.

Sixthly, during turn of client, the client can send message of using card. Server should respond corresponding result of card to sender and should inform other clients effect of using card. If a client is defeated, server would modify player status of the client to ‘defeated’. If a client becomes ‘immune’ player status, server would update player status. If client is ‘immune’ at the beginning of the turn, server would modify its player status to ‘regular’. Then server broadcasts updated client information to clients, at the same time, turn of the client is over.

Next, after a turn is over, server should check if client win the round or game and check if cards available. If no cards are available, server should notify clients then it compares card strength of survival clients and decides winner. If winner of round is decided, server should add one to player score of winner, set player status of all clients to ‘regular’ and remove cards which clients hold. Besides, card status of cards are changed to ‘regular’. If player score of a client equal to target score, it means that the client is winner of game, server would set player score of clients to 0, set player status of client to ‘regular’, remove cards which clients hold and modify card status of cards to ‘regular’. After all that, server sends winner information and updated client information to all clients. If no client reaches target score, server starts new round immediately. If winner of game is found, game would be over.

2.1.4 Commands

This part describes commands and their format which used to communicate between clients and server.

ESTABLISHED
When client connects to server successfully, server would inform client by ‘ESTABLISHED:’. However, the command only means that connection is established, the connection may be closed for some reasons.

SORRY
If client receives ‘ESTABLISHED:’, but server has been connected by 4 clients or game has been started, server would send ‘SORRY:’ to notify the client that connection can not be established.

USRNAME & EXIST & WEL
Client receives ‘ESTABLISHED:’ command, then it sends unique username to server. The command format is ‘USRNAME:[username]’. If [username] exist or invalid, then server would respond with ‘EXIST:’ command to remind client to change username until username

is valid and unique. Otherwise, server would respond ‘WEL:’ to notify client that connection is established.

USRCODE
Server uses the command to broadcasts user codes, usernames, player score and player statuses of clients which connect to server currently. Thus, clients can know each other. If there are more than one client connects to server, those data would be separated by ‘+’. Command format which server send to all clients is ‘USRCODE:[username]-[user code]-[player status]-[player score]+ [username]-[user code]-[player status]-[player score]+……’.

QUIT
Client can quit before or after game by ‘QUIT:’.

READY
Client uses the command to inform server that client is ready. The message format is ‘READY:[user code]’, [user code] belongs to sender. After receiving ready code, server updates player status of sender to ‘ready’ status then broadcasts the original command to all clients. Client displays new player statuses to users.

SCORE? & SCORE & SCORE!
The commands are used to set target score. When all clients are ready, server would send command ‘SCORE?:’ to random client for initialising target score and client responses with ‘SCORE:[target score]’. If [target score] is out of range, server would use command ‘SCORE!:’ to notify client until target score is selected appropriately.

START
When target score is initialised, server uses the command to notify clients that game is started and send target score to clients. The command format is ‘START:[target score]’. When client receives the command, it would display [target score].

CARD & TURN
When game is started, server shuffles cards and discard one card randomly each round. Then server assigns cards to clients one by one. The client is assigned first is the winner of last round. If it is first round, server would select random client as first assigned client. Card code, card name, card strength and card description are translated to client. The message format is ‘CARD: [card code]-[card name]-[card strength]-[card description]’. After sending message, card statuses of cards which are assigned to clients are updated to ‘discarded’.

Each turn of client, server use command ‘CARD’ to assign client a new card. Then, server broadcasts command ‘TURN:[user code]’ to other clients. The objective of the command is to tell clients that it is turn of client with [user code].

STRENGTH

However, if card is not available to assign, it means that game can be ended. According to rules, server compares card strength of all survival clients and find the winner who has maximum card strength. Server broadcasts card strengths of clients by command ‘STRENGTH: [user code]-[card strength]+[user code]-[card strength]+……’

WINNER
If server detects winner, server would update player score of winner and broadcast data of winner to clients. Message format is ‘WINNER:[user code]-[player score]-[card name]’. Client demonstrates username and player score of winner and card name which winner holds. Usage of command ‘WINNER’ represents the round is over.

FWINNER
If server detects player score of client equal to target score, server would inform clients by command ‘FWINNER:[user code]’. Usage of command ‘FWINNER’ means that game is over.

SKILL
When client wants to use card in other clients, the client would use command ‘SKILL’ to inform server.

‘SKILL’ command format is ‘SKILL:[user code]-[card code]-[user code]’. First user code belongs to sender, card code stands for card which sender uses and second user code belongs to target client.

However every card has unique skill, ‘SKILL’ command of cards may be different. The detailed ‘SKILL’ commands are listed at Table 4.

Used card	Command	Explanation
Guard	SKILL:[user code]-[card code]-[user code]-[card name]	First [user code] is sender, [card code] is the card, second [user code] is target client. [card name] is guess of sender. Sender can not be the target client.
Priest	SKILL:[user code]-[card code]-[user code]	First [user code] is sender, [card code] is the card, second [card code] is target client. Sender can not be the target client.
Baron	SKILL:[user code]-[card code]-[user code]	First [user code] is sender, [card code] is the card, second [card code] is target client. Sender can not be the target client.
Handmaiden	SKILL:[user code]-[card code]	First [user code] is sender, [card code] is the card.
Prince	SKILL:[user code]-[card code]-[user code]	First [user code] is sender, [card code] is the card, second [card code] is target client. Sender can not be the target client.
King	SKILL:[user code]-[card code]-[user code]+[user code]	First [user code] is sender, [card code] is the card, second and third [card code] are target clients. Sender can not be the target client.
Countess	SKILL:[user code]-[card code]	First [user code] is sender, [card code] is the card.
Princess	SKILL:[user code]-[card code]	First [user code] is sender, [card code] is the card.

Table 4

RESPONSE

When server gets ‘SKILL’ command, it would send response to client. Response is the result of using card. The command format is ‘RESPONSE:result’.

Due to every ‘SKILL’ command may has distinct response, the detailed ‘RESPONSE’ commands are listed at Table 5.

Used card	Command	Explanation
Guard	RESPONSE:	Response is unnecessary
Priest	RESPONSE:[user code]-[card name]	Response is card name of target client. First user code is the target client, card name is the card which target client has.
Baron	RESPONSE:	Response is unnecessary
Handmaiden	RESPONSE:	Response is unnecessary
Prince	RESPONSE:	Response is unnecessary
King	RESPONSE:	Response is unnecessary
Countess	RESPONSE:	Response is unnecessary
Princess	RESPONSE:	Response is unnecessary

Table 5

EFFECT
After using card, clients need to know information of card and effect of using card. The information is broadcasted to clients by server. The command format is ‘EFFECT:effect’. ‘effect’ is meaningful string, it can be displayed to users without processing.

But effect of each card is different, Table 6 illustrates detailed ‘EFFECT’ commands.

Used card	Command	Explanation
Guard	1.EFFECT:[username] uses Guard to guess [username] is [card name], it is right and [username] is defeated. 2.EFFECT:[username] uses Guard to guesses [username] is [card name], it is wrong.	First [username] is the user of card, second and third [username] is target client. [card name] is guess of user of card. Because user of card may be right or wrong, there are two kinds of command.
Priest	EFFECT:[username] uses Priest to see card of [username]	First [username] is the user of card, second [username] is target client.
Baron	EFFECT:[username] uses Baron to compare strength with [username]. [Username] is defeated.	First [username] is the user of card, second [username] is target client. Third [username] is client with lower strength.
Handmaiden	EFFECT:[username] uses Handmaiden, [username] can not be affect until next turn.	Two [username] is user of card.
Prince	1.EFFECT:[username] uses Prince to discard [card name] of [username] and [username] draw new card. 2.EFFECT:[username] uses Prince to discard [card name] of [username]. The card is Princess, [username] is defeated.	First [username] is the user of card, second and third [username] are target client, [card name] is the card which is discarded by user of card. If the card if princess, the target client is defeated, else the target client draws a new card.
King	EFFECT:[username] uses King to exchange cards of [username] and [username].	First [username] is the user of card, second and third [username] are two different target client.
Countess	EFFECT:[username] uses Countess.	
Princess	EFFECT:[username] uses Princess, [username] is defeated.	[username] is user of card.

Table 6

DEFEATED
When server finds that client is defeated, it would uses the command to inform all clients. The command format is ‘DEFEATED: [user code]’.

REGULAR
Server uses the command to modify the player status of client. The command format is ‘REGULAR:[user code]’.

IMMUNE
Server uses the command to modify the player status of client. The command format is ‘IMMUNE:[user code]’.

2.4 Connection Establishment

Connection is established by IP and TCP. Client uses IP address and port number of server to connect server. Client which receives ‘ESTABLISHED’ command connects server successfully. However, server may be full or game has been started, in the situation, connection would be closed. Client which receives ‘WEL’ command can communicates with server and the connection stays alive. Server creates thread for every client for listening requests of clients. In addition server uses single thread to send commands to clients.

2.5 Error Recovery

TCP guarantees that commands can be received by client and server. Although messages may be delay, there is no timer for timing turn of client, clients can take time to wait the commands. However, connection between server and client may be broken for some reason. In the situation, server treats client which lose connection as defeated.

2.3 Scenario

connect to server //client tries to connect to server.

server: ESTABLISHED: //connection is established.

client: USRNAME:Bob //client sends username.

server: WEL: //client creates username successfully.

server: USRCODE:Bob-0-1—0 //server broadcasts information of clients. The command means that client is Bob, user code of Bob is 0, the client is regular status(1) and player score is 0.

client(Bob): READY:0 //Bob is ready.

server: READY:0 //server broadcasts player status of Bob.

connect to server //new client tries to connect to server.

server: ESTABLISHED: //connection is established.

client: USRNAME:Bob //client sends username.

server: EXIST: //Bob is exist.

client: USRNAME:Tom //client sends new username.

server: WEL: //client creates username successfully.

server: USRCODE:Bob-0-2—0+Tom-1-1-0 //server broadcasts client information.

client(Tom): READY:1 //Tom is ready.

server: READY:1 //server broadcasts that Tom is ready.

server: SCORE?: //server ask Tom for target score.

client(Tom): SCORE:8 //Tom sets target score to 8.

server: SCORE!: //8 is out of range of target score.

client(TOM): SCORE:3 //Tom sets target score to 3.

server: START:3 //server notifies clients that game starts and tell them target score is 3.

server: CARD:1_10-guard-1-You can choose any player to guess card, if you are right, the target player lose, else nothing happen. //server selects Bob as the client who starts round.

server: TURN:0 //server broadcasts the client who starts round.

server: CARD:8_80-princess-8-You must protect this card, try to keep the card to end of game. //server distributes princess card to Tom.

server: TURN:1 //it is turn of Tom.

server: CARD:4_40-handmaiden-4-You can not be effected until your next turn //server distributes handmaiden to Bob.

server: TURN:0 //it is turn of Bob.

client(Bob): SKILL:0-1_10-1-princess //Bob uses guard to guess that Tom is princess.

server: RESPONSE: //server responses Bob.

server: EFFECT:Bob uses Guard to guess Tom is princess, it is right and Tom is defeated. //server broadcasts effect of using guard.

server: DEFEATED:1 //server broadcasts Tom is defeated.

server: WINNER:0 //server broadcasts that winner is Bob.

server: USRCODE:Bob-0-1-1+Tom-1-1-0 //server broadcasts situation of game.

If server decides that winner has to be selected by comparing card strength of clients and final winner is decided, the scenario is described follow.

server: STRENGTH:0-4+1-7 //Bob has 4 card strength and Tom has 7 card strength.

server: WINNER:1 //Tom wins the round.

server: FWINNER:1 //Tom is the final winner.

server: USRCODE:Bob-0-1-0+Tom-1-1-0 //server initialises data of clients and broadcasts them.

3. Reference

3.1 Relative RFCs

[1] Daigle, et al., “RFC Streams, Headers, Boilerplates”. RFC 5741, December 2009

[2] Information Sciences Institute, “INTERNET PROTOCOL”, RFC 793, September 1981

[3] Information Sciences Institute, “TRANSMISSION CONTROL PROTOCOL”, RFC 791, September 1981

4.Author’s Address

Shan Han
The George Washington University
2135 F St. NW
Washington DC 20052
Email: sgshan3@gwu.edu