Rochester Institute of Technology
Department of Computer Science
4003-420/4005-740 -- Data Communications and Networking I

Winter 2004-05 Course Project
Computer Network Realization of the Checkers Game

Part I

Recommended reading:

J. F. Kurose and K.W.Ross Computer Networking, Chapters 2 and 3, especially sections: 2.1.2, 2.7, 2.8, 2.9, 3.2, 3.3, 3.4

General Contents

1. You have to design, develop (write a code), run, test and evaluate the software for 2-players Checkers Game between the users according to the specifications given below. You have to develop an electronic network version of the game whose rules description is attached.
2. You have to try to make your product faster and more reliable. Although addressing security issues is not required in part 1, you might like to start thinking about making it more secure while working on part 1 also.
3. You have to develop and submit a proper documentation (see project report section).
4. You have to document :
   a. the product itself and describe how to use it,
   b. testing, you performed, and how you evaluated the product based on test results,
   c. the development process (for group projects only): how you distributed the workload, who did what, how much time you spent on different aspects, etc.

User requirements:

Please, note that requirements given below are formulated as user requirements, i.e. given by a non-expert in software engineering or computer science. These requirements could be a bit fuzzy and vague in some parts. You have to develop an exact specification for your project based on these requirements.

1. The software should be multi-game, meaning that multiple games may try to get an access to the game-server at the same time.
2. Each game should have a unique game name, which will be formed by concatenating of the players names. Games between the same players or two

games by the same player are not allowed simultaneously, i.e. game Allyson-Bob and Bob-Allyson are considered the same and not allowed.

3. Developing of a nice GUI is not required and generally will not be included into assessment but you might get some bonus points for outstanding solutions. You are allowed to use GUI developed in Software Engineering and other courses.

Software Requirements

Overall Requirements

1. The software must be written in Java. It must run on CS Lab computers without any additional software being installed. Note that CS computers were recently upgraded with an installation of a new Java version.
2. The Java classes for the software must not be in a package. (That is, no **package x.y.z;** statements.)
3. The program must use client-server architecture with a server process and a process for each user.
4. You have to implement both TCP and UDP protocols in your project.
5. The software will use sockets (TCP) and datagrams (UDP) to communicate between users and the server.

You have to continue this list yourself and develop further requirements as a part of specifications.

Server-Client Protocol Requirements

1. A user connects to the server by sending the following information:
   a. Protocol type (TCP or UDP) and port to connect to.
   b. Player Name (the Player Name is one or more characters and may not contain blanks)
   c. Player Name of the person, he/she wants to play with, in the same format as in b.
2. Note that players participating in the same session may request communicating in the same protocol format or in different formats (one –TCP, another –UDP) that is a legal request, which should be accommodated in your project.
3. If the request to start game is legal (no such session on the server), the server starts session between these users by sending following to one or both users:
   a. Player Name
   b. Ready or not ready to communicate. If a first player has made a request and a second one has not yet, the server informs the first player to wait until the second user issues the same request. If a second player made a request, then the server informs both users that they are ready to communicate.
4. If a request to play is illegal (a session between the same players is active or one of the user has already made a request to play with another player), the server does not start a new session; instead, the server sends an error message to the player.
5. When game is over server sends message to the players with the name and the current checkers positions, and then disconnects them from the server.
6. The protocol must use a textual format for a TCP client and a binary format (be read or written using binary input or output) for a UDP client for each message.

Server Protocol Requirements

1. The server must support multiple simultaneous game sessions.
2. The server must keep track of the state of each separate session.
3. For each session, the server must perform the server side of the server-client protocol specified above.
4. When a user requests to play in a session that does not exist, the server must create the session.
5. When a user requests to play and the request is legal, the server must verify if both partners are ready and inform them applying protocols users are utilizing. That means that players may use different protocols to communicate, and server must perform accordingly.
6. The server process **must** be run by typing this command line:

```
java Server <TCPport> <UDPport>
```

where

<TCPport> is the port number on which the server is accepting socket connections

<UDPport> is the port number of the server's datagram connections.

NOTE: This means that the server main program class **must** be named Server, and this class must not be in a package. The violation of this convention will be penalized.

7. The server process must continue running until externally stopped.

Client Protocol Requirements

1. The player can participate in one game session only at the same time.
2. The player must perform the client side of the server-client protocol specified above.
3. When a client receives the server's response, it must output the following line on the standard output:

- If it is an initial request and the server is not ready to establish play session:

  `Game with <OtherPlayer> is not available. Hold on ...`

  where <OtherPlayer> is a name of the partner

- If the server is ready to establish a play session:

  `Game with <OtherPlayer> is established ...`

  where <OtherPlayer> is the partner's name.

4. When a player receives an error message from the server in response to a game request, the player must print the error message on the standard output.

5. A player must keep track of the checkers positions and output them on request.
6. After the 15 seconds waiting time, a player generates next move and sends it to the server along with the checkers positions
7. When a player receives a winning message from the server, it outputs it along with the checkers positions
10. The above-specified printouts are the only printouts a player may output.
11. A player must exit after receiving the end of the game message from the server.
12. The player's process **must** be initiated by typing this command line:

    ```
    java Player <host> <protocol> <port> <yourName>
    <otherName>
    ```

    where

    <host> is the name of the host computer where the server is running,

    <protocol> is the protocol type, "TCP" or "UDP" only (all capital letters),

    <port> is the port number on which the server is accepting socket connections or the server's datagram mailbox,

    <yourName> is the Player name for this game session, and

    <otherName> is the partner name for this session.

NOTE: This means that the player main program class **must** be named Player, and this class must not be in a package. The violation of this convention will be penalized.

Submission Requirements for Project 1.

(1) Submission deadline is 11.59 pm, Wednesday, January 12, 2005
(2) Late submission or resubmissions will be penalized by 10% grade reduction per day for group projects and 5% for individual projects. Note that a delay longer than one hour after midnight will be counted as a day.

(3) Please, submit your project by email to icss420@cs.rit.edu

You have to submit:

a file named pr1_yourname.* should contain your documentation in doc, txt or pdf formats.

a file named pr1_yourname.jar should contain the Java source file for all classes in your program. Each source file should include Javadoc comments describing the class or interface. Each method within each class or interface must include Javadoc comments describing the overall method, the arguments if any, the return value if any, and the exceptions thrown if any. Apply the command jar cvf pr1_yoursurname.jar *.java to create this file. These two files should be submitted by the deadline.

Documentation

Suggested report format

Executive summary (1-2 paragraphs)

Concise description of problem addressed and results

Requirements (1-2 paragraphs)

Your brief understanding of what the instructor requires in this project - informally stated

Specification (1-2 pages)

Precise definition of what you are to do and what results you have to achieve. You might like to formulate it as protocol requirements, separating them into user and server parts. Please, be as specific as possible and provide as much detail as you can here.

Feasibility study (1 page) **(for group projects only)**

Possible solutions, protocols, models and methods

Comparison (at least three comparisons of different options) of solutions (you have to compare at least two in each comparison, e.g. TCP vs. UDP is one example, but you should NOT include it into your study)

Conclusion:

Choose one of each and explain why

Implementation (1-2 pages)

Structure, contents, user interface, limitations, software and hardware requirements, etc. Describe all the classes applied. If you used a code from some library, provide the reference to this library.

Product testing (2-3 pages)

Compilation:

- State the time of your final compilation, hardware and software environment, messages (if any) received and the result of your compilation.

Testing:

-Actual executed test cases must be described briefly (you should run and describe at least 10 test cases). Briefly describe how you designed your tests: your main concept and ideas, why you tested that and this, what you expected

- The results of your tests
- Evaluation of your product performance and reliability based on your test results

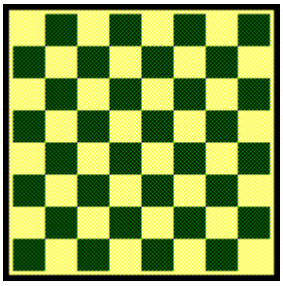
Development process documentation (1-2 pages for group projects only):

Describe the process of the software development and testing; briefly describe who did what and how your team work was organized and approximately how much time you spent on different project components.

CHECKERS

A Two Player Game

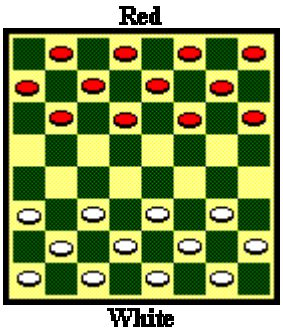
Here are very basic rules, which I found at http://www.jimloy.com/checkers/rules2.htm
You can use another version if you prefer.



Checkers is a board game played between two players, that take alternate moves. The player cannot move, when he has no pieces, or his pieces are blocked, hence loses the game. Players can resign or agree to draw.

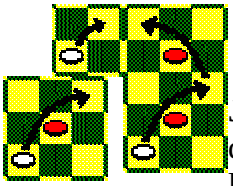
The board is square, with sixty-four smaller squares, arranged in an 8x8 grid. The smaller squares are alternately light and dark colored (green and buff in tournaments), in the famous "checker-board" pattern. The game of checkers is played on the **dark** (black or green) squares. Each player has a dark square on his far left and a light square on his far right. The double-corner is the distinctive pair of dark squares in the near right corner.

The pieces are Red and White, and are called Black and White in most books. In some modern publications, they are called Red and White. Sets bought in stores may be other colors. Black and Red pieces are still called Black (or Red) and White, so that you can read the books. The pieces are of cylindrical shape, much wider than they are tall (see diagram). Tournament pieces are smooth, and have no designs (crowns or concentric circles) on them. The pieces are placed on the **dark** squares of the board.



The starting position is with each player having twelve pieces, on the twelve dark squares closest to his edge of the board. Notice that in checker diagrams, the pieces are usually placed on the light colored squares, for readability. On a real board they are on the dark squares.

Moving: A piece which is not a king can move one square, diagonally, forward, as in the diagram at the right. A king can move one square diagonally, forward or backward. A piece (piece or king) can only move to a vacant square. A move can also consist of one or more jumps (next paragraph).



Jumping: You capture an opponent's piece (piece or king) by jumping over it, diagonally, to the adjacent vacant square beyond it. The three squares must be lined up (diagonally adjacent) as in the diagram at the left: your jumping piece (piece or king), opponent's piece (piece or king), empty square. A king can jump diagonally, forward or backward. A piece which is not a king, can only jump diagonally forward. You can make a

multiple jump (see the diagram on the right), with one piece only, by jumping to empty square to empty square. In a multiple jump, the jumping piece or king can change directions, jumping first in one direction and then in another direction. You can only jump one piece with any given jump, but you can jump several pieces with a move of several jumps. You remove the jumped pieces from the board. You cannot jump your own piece. You cannot jump the same piece twice, in the same move. If you can jump, you must. And, a multiple jump must be completed; you cannot stop part way through a multiple jump. If you have a choice of jumps, you can choose among them, regardless of whether some of them are multiple, or not. A piece, whether it is a king or not, can jump a king.

Kinging: When a piece reaches the last row (the King Row), it becomes a King. A second checker is placed on top of that one, by the opponent. A piece that has just kinged, cannot continue jumping pieces, until the next move.

Red moves first. The players take turns moving. You can make only one move per turn. You must move. If you cannot move, you lose. Players normally choose colors at random, and then alternate colors in subsequent games.