





ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN
CƠ SỞ NGÀNH MẠNG

GIÁO VIÊN HƯỚNG DẪN : TRẦN HỒ THỦY TIÊN
SINH VIÊN TỰC HIỆN: TRƯƠNG QUAN HÙNG
ĐOÀN VĂN HOÀNG

Đà Nẵng, 12/2020



NHÓM 6:

Điểm	Sinh viên	Phân công	Chữ ký sinh viên	Chữ ký giáo viên
	Trương Quang Hùng	Phần hệ điều hành: tìm tài liệu, code giải thuật. Phần mạng: tìm tài liệu, viết báo cáo, code phía client, kiểm tra lỗi.		
	Đoàn Văn Hoàng	Phần hệ điều hành: tìm tài liệu, code giao diện, viết báo cáo, kiểm tra lỗi. Phần mạng: tìm tài liệu, viết báo cáo, code phía server, chạy thử.		

TÓM TẮT:

Phần hệ điều hành: Nghiên cứu về vấn đề deadlock trong hệ điều hành và áp dụng thuật toán nhà băng (banker) của Dijkstra giúp phát hiện khả năng xảy ra deadlock. Phần mạng tìm hiểu về giao thức TCP/IP để xây dựng chương trình chơi cờ Đam (checker) với ngôn ngữ lập trình java.

TỪ KHÓA: Deadlock, Banker Algorithm, Dijkstra, TCP/IP, Checker.

MỤC LỤC

MỤC LỤC.....	2
LỜI CẢM ƠN	3
PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH.....	4
I. TỔNG QUAN VỀ ĐỀ TÀI.....	4
1. Bối cảnh và lý do chọn đề tài.....	4
II. CƠ SỞ LÝ THUYẾT.	4
1. Giới thiệu về Deadlock.	4
2. Thuật toán nhà băng của Dijkstra.....	6
III. THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH	7
1. Phân tích yêu cầu.....	7
2. Xây dựng chương trình.....	7
3. Kết quả thử nghiệm.....	8
IV. ĐÁNH GIÁ KẾT QUẢ.....	10
1. Hạn chế của thuật toán.....	10
2. Kết luận.....	10
PHẦN II: LẬP TRÌNH MẠNG.....	11
I. CƠ SỞ LÝ THUYẾT.....	11
1. Giao thức TCP/IP.....	11
2. Lập trình Socket và cổng port.	13
3. Mô hình Client/Server.....	13
II. PHÂN TÍCH THIẾT KẾ HỆ THỐNG.	15
1. Phân tích yêu cầu.....	15
2. Phân tích hệ thống.....	16
3. Kết quả.....	18
III. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	20
1. Kết luận.....	20
2. Hướng phát triển	20
TÀI LIỆU THAM KHẢO	21

LỜI CẢM ƠN

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian thực hiện đề tài, nhóm em đã nhận được rất nhiều sự quan tâm giúp đỡ của quý thầy/cô, gia đình và bạn bè.

Với lòng biết ơn sâu sắc chúng em xin gửi đến quý thầy cô ở khoa Công nghệ thông tin đã cùng với tri thức và tâm huyết của mình truyền đạt vốn kiến thức quý báu cho nhóm em. Và đặc biệt trong học kỳ này, khoa đã giúp cho em tiếp cận được với môi trường làm việc thực tế thông qua học phần “Đồ án cơ sở ngành mạng”. Để hoàn thành được đồ án môn học này, em xin chân thành cảm ơn đến Cô Trần Hồ Thủy Tiên đã tận tình giúp đỡ nhóm em trong suốt thời gian làm đồ án. Sự thành công của môn học này chính là nhờ sự đóng góp công sức không hề nhỏ của thầy/cô. Trong quá trình hoàn thành công việc, chúng em không thể tránh được sai sót. Vậy nên, nhóm em rất mong quý thầy/cô thông cảm cho những sai sót ấy và ghi nhận những gì cả nhóm đã làm được.

Một lần nữa, nhóm em xin cảm ơn quý thầy/cô đã bỏ ra thời gian quý báu của mình để thông qua đồ án Cơ sở ngành mạng của em.

Sau cùng, chúng em xin kính chúc quý thầy/cô thật dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ mai sau.

Xin chân thành cảm ơn!

Đà Nẵng, ngày 19 tháng 12 năm 2020

Nhóm Sinh viên thực hiện

Nhóm 6:

Trương Quang Hùng

Đoàn Văn Hoàng

PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH.

I. TỔNG QUAN VỀ ĐỀ TÀI.

1. Bối cảnh và lý do chọn đề tài.

Hệ điều hành (Operating System - OS) là chương trình quản lý phần cứng và các tài nguyên phần mềm trên máy tính. Hệ điều hành đóng vai trò trung gian trong việc giao tiếp giữa người sử dụng và phần cứng máy tính, cung cấp một môi trường cho phép người sử dụng phát triển và thực hiện các ứng dụng của họ một cách dễ dàng.

Trong môi trường xử lý đa chương, nhiều quá trình được xử lý đồng thời để tối ưu hóa việc sử dụng CPU. Một quá trình yêu cầu tài nguyên, nếu tài nguyên không sẵn có thì quá trình đi vào trạng thái chờ. Nếu tài nguyên này bị giữ bởi những trạng thái chờ khác thì quá trình đó sẽ bị khóa chết (gọi là deadlock).

Đối với một hệ thống lớn, bao gồm nhiều quá trình, chương trình đa luồng, nhiều tài nguyên thì giải quyết deadlock thực sự là một vấn đề rất quan trọng và không thể bỏ qua. Muốn giải quyết được Deadlock cần nắm vững kiến thức về lý thuyết của hệ điều hành và các giải thuật liên quan. Do đó, khi xử lý thành công Deadlock, chúng ta sẽ hiểu cặn kẽ môn học Nguyên Lý Hệ Điều Hành.

Mục tiêu của đề tài:

- Hiểu mô hình hệ thống về deadlock.
- Hiểu các đặc điểm của deadlock.
- Hiểu các phương pháp quản lý deadlock.
- Hiểu cách ngăn chặn deadlock.
- Hiểu cách tránh deadlock.
- Hiểu cách phát hiện deadlock.
- Hiểu cách phục hồi từ deadlock.

Yêu cầu của đề tài:

- Tìm hiểu Deadlock.
- Trình bày thuật toán Banker.
- Xây dựng chương trình và kết quả demo.
- Ngôn ngữ dùng để viết chương trình: java.

II. CƠ SỞ LÝ THUYẾT.

1. Giới thiệu về Deadlock.

1.1 Tiến trình.

1.1.1 Định nghĩa.

- Tiến trình (process) là trạng thái tức thời của một chương trình đang chạy trên máy tính. Nó bao gồm bộ nhớ cần thiết để chạy chương trình (không gian địa chỉ

của tiến trình) và khả năng kiểm soát hiện trạng của bộ xử lý trong tiến trình thực thi chương trình.

- Được xem là đơn vị làm việc trong các hệ điều hành.
- Là một thực thể chủ động, khác với chương trình là một thể bị động.

1.2 Deadlock

1.2.1 Giới thiệu.

- Trong quá trình lập lịch tiến trình, quá trình chờ của các tiến trình có thể không bao giờ chuyển trạng thái trở lại vì tài nguyên chúng yêu cầu bị giữ bởi những tiến trình đang chờ khác. Trường hợp này được gọi là deadlock.

1.2.2 Đặc điểm của Deadlock.

Những điều kiện cần thiết gây ra deadlock:

Trường hợp deadlock có thể phát sinh nếu bốn điều kiện sau xảy ra cùng một lúc trong hệ thống:

1. *Loại trừ hồ tương:* ít nhất một tài nguyên phải được giữ trong chế độ không chia sẻ; nghĩa là, chỉ một quá trình tại cùng một thời điểm có thể sử dụng tài nguyên. Nếu một quá trình khác yêu cầu tài nguyên đó, quá trình yêu cầu phải tạm dừng cho đến khi tài nguyên được giải phóng.
 2. *Giữ và chờ cấp thêm tài nguyên:* quá trình phải đang giữ ít nhất một tài nguyên và đang chờ để nhận tài nguyên thêm mà hiện đang được giữ bởi quá trình khác.
 3. *Không đòi lại tài nguyên từ quá trình đang giữ chúng:* Các tài nguyên không thể bị đòi lại; nghĩa là, tài nguyên có thể được giải phóng chỉ tự ý bởi quá trình đang giữ nó, sau khi quá trình đó hoàn thành tác vụ.
 4. *Tồn tại chu trình trong đồ thị cấp phát tài nguyên:* một tập hợp các quá trình $\{P_0, P_1, \dots, P_n\}$ đang chờ mà trong đó P_0 đang chờ một tài nguyên được giữ bởi P_1 , P_1 đang chờ tài nguyên đang giữ bởi P_2, \dots, P_{n-1} đang chờ tài nguyên đang được giữ bởi quá trình P_0 .
- Chúng ta nhấn mạnh rằng tất cả bốn điều kiện phải cùng phát sinh để deadlock xảy ra. Điều kiện chờ đợi chương trình đưa đến điều kiện giữ-và-chờ vì thế bốn điều kiện không hoàn toàn độc lập.

1.3 Các phương pháp xử lý Deadlock.

- Có 3 cách giải quyết deadlock cơ bản, đó là:
 1. Chúng ta có thể sử dụng một giao thức để ngăn chặn hay tránh deadlocks, đảm bảo rằng hệ thống sẽ không bao giờ đi vào trạng thái deadlock .
 2. Chúng ta có thể cho phép hệ thống đi vào trạng thái deadlock, phát hiện nó và phục hồi.

3. Chúng ta có thể bỏ qua hoàn toàn vấn đề này và giả vờ deadlock không bao giờ xảy ra trong hệ thống. Giải pháp này được dùng trong nhiều hệ điều hành, kể cả UNIX.

- Ở đề tài này sẽ chỉ nói về các thuật toán tránh deadlock và chi tiết về thuật toán nhà băng của Dijkstra.

2. Thuật toán nhà băng của Dijkstra.

2.1 Giới thiệu.

- Là một thuật toán cấp phát tài nguyên và tránh deadlock được phát triển bởi Dijkstra.
- Áp dụng cho hệ thống có nhiều loại tài nguyên.
- Thuật toán kiểm tra trạng thái an toàn bằng cách mô phỏng phỏng việc cấp phát lượng tối đa có thể của tất cả các loại tài nguyên được xác định trước, sau đó kiểm tra trạng thái an toàn xem điều kiện xảy ra deadlock của tất cả các tiến trình đang yêu cầu tài nguyên có thể xảy ra không. Sau đó dựa vào trạng thái an toàn hay không để quyết định việc cấp phát tài nguyên.

2.2 Cấu trúc dữ liệu.

- Nhiều cấu trúc dữ liệu phải được duy trì để cài đặt giải thuật Banker. Những cấu trúc dữ liệu này mã hoá trạng thái của hệ thống cấp phát tài nguyên. Gọi n là số quá trình trong hệ thống và m là số loại tài nguyên trong hệ thống. Chúng ta cần các cấu trúc dữ liệu sau:

- **Available:** một vector có chiều dài r hiển thị số lượng tài nguyên sẵn dùng của mỗi loại. Nếu $Available[j] = k$, có k thể hiện của loại tài nguyên R_j sẵn dùng.
- **Max:** một ma trận $P \times R$ định nghĩa số lượng tối đa yêu cầu của mỗi quá trình. Nếu $Max[i][j] = k$, thì quá trình P_i có thể yêu cầu nhiều nhất k thể hiện của loại tài nguyên R_j .
- **Allocation:** một ma trận $P \times R$ định nghĩa số lượng tài nguyên của mỗi loại hiện được cấp tới mỗi quá trình. Nếu $Allocation[i][j] = k$, thì quá trình P_i hiện được cấp k thể hiện của loại tài nguyên R_j .
- **Need:** một ma trận $P \times R$ hiển thị yêu cầu tài nguyên còn lại của mỗi quá trình. Nếu $Need[i][j] = k$, thì quá trình P_i có thể cần thêm k thể hiện của loại tài nguyên R_j để hoàn thành tác vụ của nó. Chú ý rằng, $Need[i][j] = Max[i][j] - Allocation[i][j]$.

2.3 Thuật toán kiểm tra an toàn.

- Giải thuật để xác định hệ thống ở trạng thái an toàn hay không có thể được mô tả như sau:

Bước 1: Gọi Work và Finish là các vector có chiều dài m và n tương ứng. Khởi tạo $Work := Available$ và $Finish[i] := false$ cho $i = 1, 2, \dots, n$.

Bước 2: Tìm i thỏa:

- a) $Finish[i] = false$
- b) $Need[i] \leq Work$. Nếu không có i nào thỏa, di chuyển tới bước 4

Bước 3: $Work := Work + Allocation[i]$

$Finish[i] := true$. Di chuyển về bước 2.

Bước 4: Nếu $Finish[i] = true$ cho tất cả i, thì hệ thống đang ở trạng thái an toàn. Giải thuật này có thể yêu cầu độ phức tạp $m \times n^2$ thao tác để quyết định trạng thái là an toàn hay không.

III. THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH .

1. Phân tích yêu cầu.

Bài toán đặt ra.

Input: Số tiến trình P, số tài nguyên R được nhập vào ô textbox ($P \leq 10, R \leq 10$).

Các giá trị tài nguyên được tạo ngẫu nhiên theo giới hạn P và R được nhập trước.

Output: Kiểm tra trạng thái an toàn của tiến trình khi yêu cầu tài nguyên, nếu an toàn thì cấp phát tài nguyên và tiếp tục yêu cầu cho đến khi các tiến trình hoàn tất.

Bảng thể hiện ma trận Work: thứ tự cấp phát tài nguyên cho các tiến trình.

2. Xây dựng chương trình.

2.1 Các biến, hàm sử dụng trong chương trình

STT	Tên hàm, biến	Chức năng
1	$Allocation[][]$	Mảng Allocation chứa tài nguyên đã cấp phát
2	$Max[][]$	Mảng Max chứa tài nguyên tối đa mà các tiến trình yêu cầu
3	$Need[][]$	Mảng Need chứa tài nguyên mà các tiến trình cần để hoàn thành
4	$Available[]$	Mảng Available chứa tổng số tài nguyên có sẵn
5	$Work[]$	Mảng Work thể hiện quá trình cấp phát tài nguyên
6	$Finish[]$	Mảng Finish để kiểm tra trạng thái hệ thống
7	R	Số tài nguyên

8	P	Số tiến trình
9	CreateData()	Khởi tạo dữ liệu ban đầu
10	Update_Q(int k)	Cập nhật lại tài nguyên sau khi 1 tiến trình hoàn thành
11	RunProcessor()	Khởi chạy hệ thống

- Các mảng trên được cài đặt đều là mảng song song.
- Chương trình được cài đặt bằng ngôn ngữ lập trình JAVA.

3. Kết quả thử nghiệm.

- Hệ thống an toàn:

ĐỒ ÁN HỆ ĐIỀU HÀNH

Xây Dựng Chương Trình Mô Phỏng Giải Thuật Nhà Bể Của Dijkstra Để Dự Đoán Deadlock

SỐ TIẾN TRÌNH P (<=10): SỐ TÀI NGUYÊN R (<=10):

BẢNG DỮ LIỆU MAX

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]
P[1]	6	2	5	9	0					
P[2]	4	2	4	0	8					
P[3]	5	5	5	3	1					
P[4]	5	1	8	5	8					

BẢNG DỮ LIỆU ALLOCATION

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]
P[1]	0	0	2	7	0					
P[2]	2	2	4	0	8					
P[3]	3	5	1	1	0					
P[4]	5	1	0	5	3					

BẢNG DỮ LIỆU NEED

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]
P[1]	6	2	3	2	0					
P[2]	2	0	0	0	0					
P[3]	2	0	4	2	1					
P[4]	0	0	8	0	5					

BẢNG DỮ LIỆU AVAILABLE

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]
	2		5	7	4	8				

BẢNG DỮ LIỆU WORK KHI THỰC THI CÁC TIẾN TRÌNH

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]
P[2]	4	7	11	4	16					
P[3]	7	12	12	5	16					
P[1]	7	12	14	12	16					
P[4]	12	13	14	17	19					

Hệ thống an toàn

Hình 1: Kết quả kiểm tra hệ thống an toàn.

+ Các tiến trình đều được cấp phát đủ tài nguyên yêu cầu. Thứ tự các tiến trình được mô tả trong bản dữ liệu Work như trong hình.

- Hệ thống không an toàn .

ĐỒ ÁN HỆ ĐIỀU HÀNH

Xây Dựng Chương Trình Mô Phỏng Giải Thuật Nhà Bể Của Dijkstra Để Dự Đoán Deadlock

SỐ TIẾN TRÌNH P (<=10): SỐ TÀI NGUYÊN R (<=10):

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]
P[1]	3	3	8	9	1					
P[2]	4	3	6	1	4					
P[3]	3	9	6	1	3					
P[4]	7	3	2	3	9					

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]
P[1]	3	2	5	2	1					
P[2]	0	3	5	0	0					
P[3]	3	0	0	1	3					
P[4]	5	0	2	0	4					

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]
P[1]	0	1	3	7	0					
P[2]	4	0	1	1	4					
P[3]	0	9	6	0	0					
P[4]	2	3	0	3	5					

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]
	2	3	4	1	3					

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]

Hệ thống không an toàn

Hình 2: Kết quả kiểm tra hệ thống không an toàn với không tiến trình nào thực thi.

+ Không tiến trình nào được cấp phát đủ tài nguyên yêu cầu dẫn đến việc hệ thống xảy ra Deadlock.

ĐỒ ÁN HỆ ĐIỀU HÀNH

Xây Dựng Chương Trình Mô Phỏng Giải Thuật Nhà Bể Của Dijkstra Để Dự Đoán Deadlock

SỐ TIẾN TRÌNH P: SỐ TÀI NGUYÊN R:

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]
P[1]	3	4	1	0	9	0	3			
P[2]	2	0	9	3	3	9	2			
P[3]	0	6	1	1	8	2	9			
P[4]	0	8	5	0	4	1	3			

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]
P[1]	3	0	0	0	8	0	2			
P[2]	2	0	6	1	2	3	2			
P[3]	0	3	1	0	4	1	5			
P[4]	0	7	2	0	1	1	3			

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]
P[1]	0	4	1	0	1	0	1			
P[2]	0	0	3	2	1	6	0			
P[3]	0	3	0	1	4	1	4			
P[4]	0	1	3	0	3	0	0			

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]
	9	8	5	3	4	3	0			

	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]
P[4]	9	15	7	3	5	4	3			
P[1]	12	15	7	3	13	4	5			
P[3]	12	18	8	3	17	5	10			

Hệ thống không an toàn

Hình 3: Kết quả kiểm tra hệ thống không an toàn.

+ Hệ thống chỉ cấp phát đủ tài nguyên cho $\frac{3}{4}$ các tiến trình nên làm cho hệ thống xảy ra Deadlock.

IV. ĐÁNH GIÁ KẾT QUẢ.

1. Hạn chế của thuật toán.

- Thuật toán banker xuất phát từ giả thiết số tài nguyên là cố định. Nhưng bởi vì các tài nguyên không thể làm việc mãi (ví dụ dừng lại để bảo dưỡng) do đó chúng ta không thể cho rằng số lượng tài nguyên là cố định.
- Thuật toán đòi hỏi rằng số người dùng là không đổi. Yêu cầu đó cũng không thực tế, vì trong các hệ đa chương trình, số lượng người dùng luôn thay đổi.
- Cũng như thế, thuật toán đòi hỏi người dùng phải trả lại các tài nguyên được cấp, sau một khoảng thời gian nào đó- và trong thực tế cũng cần các chỉ số cụ thể.
- Thuật toán yêu cầu người dùng phải báo trước số lượng lớn nhất tài nguyên anh ta cần. Nhưng sự phân phối tài nguyên ngày càng phải linh động, và do đó càng khó đánh giá yêu cầu lớn nhất. Vì máy tính ngày càng thân thiện với người dùng nên sẽ ngày càng nhiều người dùng không có hình dung chính xác về số tài nguyên lớn nhất mà anh ta sẽ cần, thay vào đó khi nào cần tài nguyên người dùng mới yêu cầu.

2. Kết luận.

Thực hiện đề tài mô phỏng thành công thuật toán Banker để tránh Deadlock, hiểu rõ hơn cách mà hệ điều hành quản lý phần cứng và các tài nguyên phần mềm trên máy tính, qua đó đã giúp chúng ta hiểu cặn kẽ hơn về môn học Nguyên Lí Hệ Điều Hành.

PHẦN II: LẬP TRÌNH MẠNG.

ĐỀ TÀI: Xây dựng chương trình chơi cờ Đam (Checker) với giao thức TCP/IP.

I. CƠ SỞ LÝ THUYẾT.

1. Giao thức TCP/IP.

1.1. Tổng quan.

TCP/IP là bộ giao thức cho phép kết nối các hệ thống mạng không đồng nhất với nhau. Ngày nay TCP/IP được sử dụng rộng rãi trong mạng cục bộ cũng như mạng toàn cầu. TCP/IP được xem như giản lược của mô hình tham chiếu OSI với 4 tầng như sau: o Tầng Liên Kết (Datalink Layer) o Tầng Mạng (Internet Layer) o Tầng Giao Vận (Transport Layer) o Tầng Ứng Dụng (Application Layer).

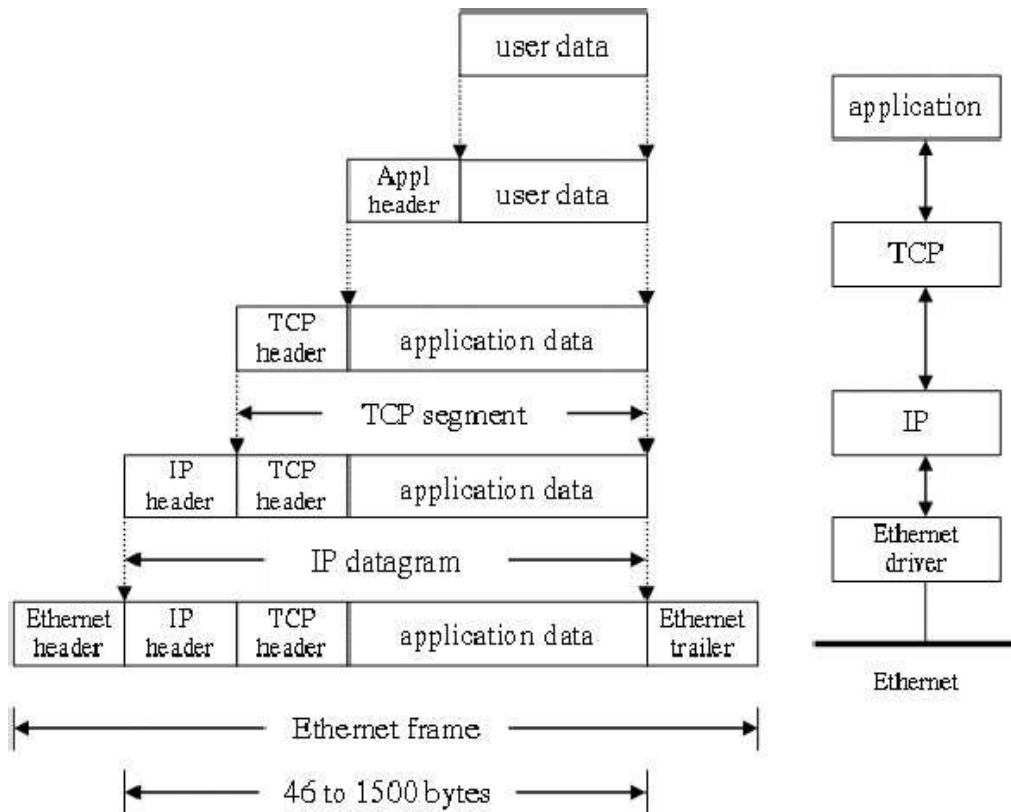
Tầng liên kết: Tầng liên kết (còn được gọi là tầng liên kết dữ liệu hay tầng giao tiếp mạng) là tầng thấp nhất trong mô hình TCP/IP, bao gồm các thiết bị giao tiếp mạng và các chương trình cung cấp các thông tin cần thiết để có thể hoạt động, truy nhập đường truyền vật lý qua các thiết bị giao tiếp mạng đó.

Tầng Internet: Tầng Internet (hay còn gọi là tầng Mạng) xử lý quá trình truyền gói tin trên mạng, các giao thức của tầng này bao gồm : IP (Internet Protocol) , ICMP (Internet Control Message Protocol) , IGMP (Internet Group Message Protocol).

Tầng giao vận: Tầng giao vận phụ trách luồng dữ liệu giữa 2 trạm thực hiện các ứng dụng của tầng trên, tầng này có 2 giao thức chính là TCP (Transmission Control Protocol) và UDP (User Datagram Protocol) - TCP cung cấp luồng dữ liệu tin cậy giữa 2 trạm, nó sử dụng các cơ chế như chia nhỏ các gói tin ở tầng trên thành các gói tin có kích thước thích hợp cho tầng mạng bên dưới, báo nhận gói tin, đặt hạn chế thời gian timeout để đảm bảo bên nhận biết được các gói tin đã gửi đi. Do tầng này đảm bảo tính tin cậy nên tầng trên sẽ không cần quan tâm đến nữa - UDP cung cấp một dịch vụ rất đơn giản hơn cho tầng ứng dụng . Nó chỉ gửi dữ liệu từ trạm này tới trạm kia mà không đảm bảo các gói tin đến được tới đích. Các cơ chế đảm bảo độ tin cậy được thực hiện bởi tầng trên Tầng ứng dụng.

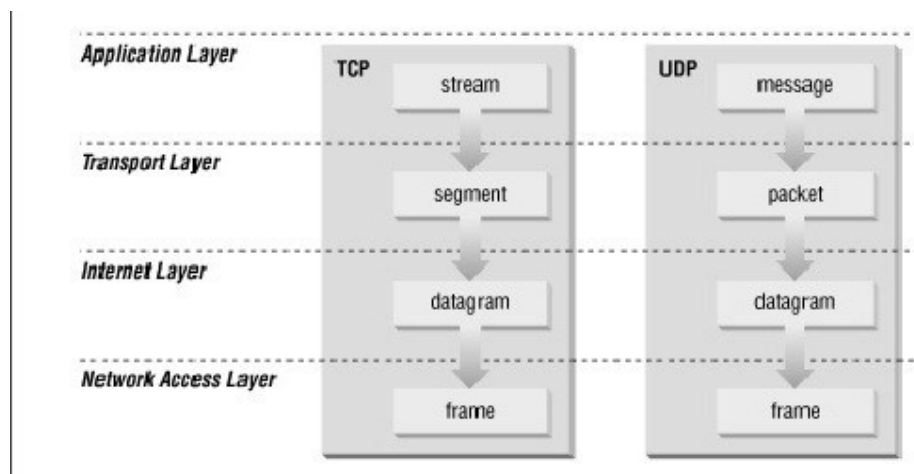
Tầng ứng dụng: là tầng trên của mô hình TCP/IP bao gồm các tiến trình và các ứng dụng cung cấp cho người sử dụng để truy cập mạng. Có rất nhiều ứng dụng được cung cấp trong tầng này , mà phổ biến là Telnet: sử dụng trong việc truy cập mạng từ xa, FTP (File Transport Protocol) dịch vụ truyền tệp tin, EMAIL : dịch vụ truyền thư tín điện tử. WWW (Word Wide Web).

1.2. Phương thức hoạt động của bộ giao thức TCP/IP.



Hình 4: Phương thức hoạt động của bộ giao thức TCP/IP

Cũng tương tự như trong mô hình OSI, khi truyền dữ liệu, quá trình tiến hành từ tầng trên xuống tầng dưới, qua mỗi tầng dữ liệu được thêm vào thông tin điều khiển gọi là Header. Khi nhận dữ liệu thì quá trình xảy ra ngược lại. dữ liệu được truyền từ tầng dưới lên và qua mỗi tầng thì phần header tương ứng sẽ được lấy đi và khi đến tầng trên cùng thì dữ liệu không còn phần header nữa.



Hình 5: Cấu trúc dữ liệu trong TCP/IP

Hình trên cho ta thấy lược đồ dữ liệu qua các tầng. Trong hình ta thấy tại các tầng khác nhau dữ liệu được mang những thuật ngữ khác nhau:

- Trong tầng ứng dụng: dữ liệu là các luồng được gọi là stream.
- Trong tầng giao vận: đơn vị dữ liệu mà TCP gửi xuống gọi là TCP segment.
- Trong tầng mạng, dữ liệu mà IP gửi xuống tầng dưới gọi là IP Datagram.
- Trong tầng liên kết, dữ liệu được truyền đi gọi là frame.

2. Lập trình Socket và cổng port.

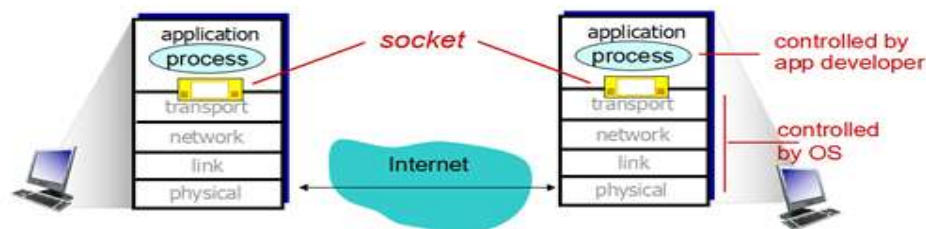
2.1. Khái niệm socket và cổng port.

2.1.1. Socket

Socket là một điểm cuối trong một kết nối giữa hai chương trình đang chạy trên mạng.

Trên quan điểm của người phát triển ứng dụng, **Socket** là một phương pháp để thiết lập kết nối truyền thông giữa một chương trình yêu cầu dịch vụ (được gắn nhãn Client) và một chương trình cung cấp dịch vụ (được gắn nhãn là Server) trên mạng hoặc trên cùng một máy tính.

Đối với người lập trình, họ nhìn nhận **Socket** như một giao diện nằm giữa tầng ứng dụng và tầng khác trong mô hình mạng OSI có nhiệm vụ thực hiện việc giao tiếp giữa chương trình ứng dụng với các tầng bên dưới của mạng.



Hình 6: Socket

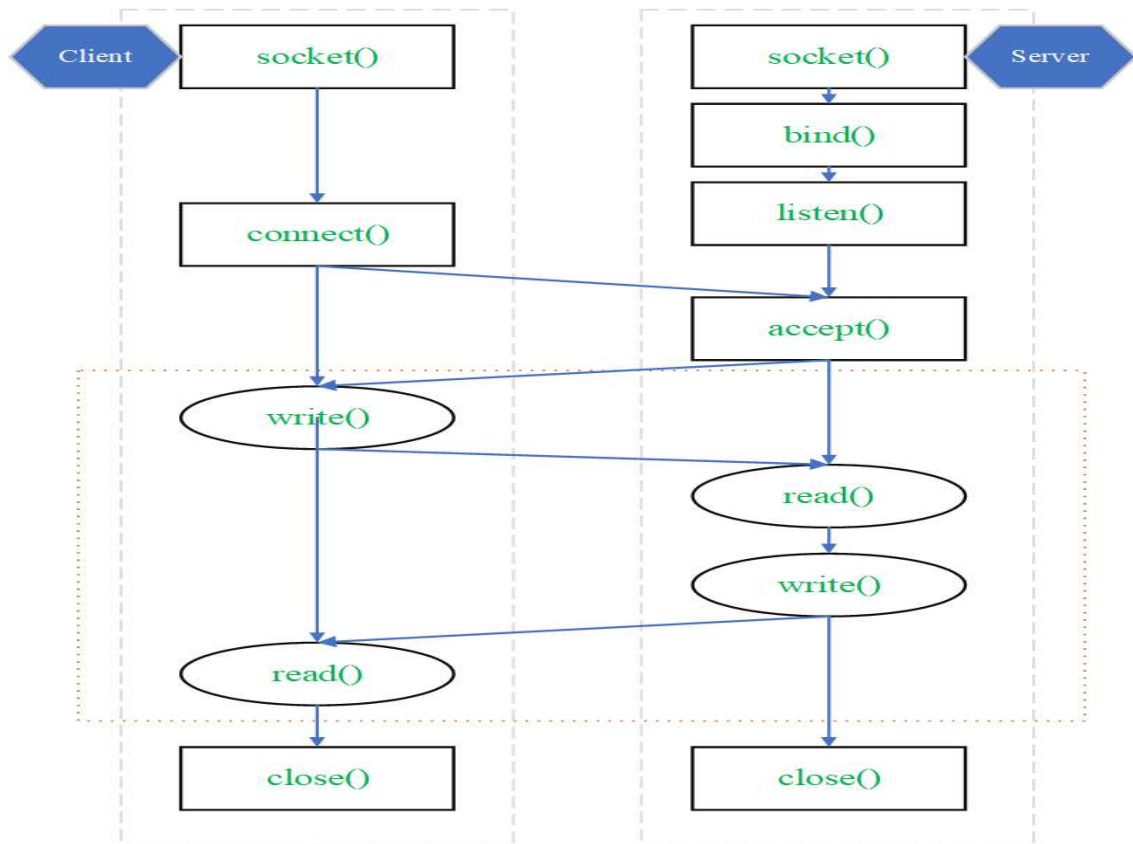
2.1.2. Số hiệu của Socket (cổng port)

Để có thể thực hiện các cuộc giao tiếp, một trong hai quá trình phải công bố số hiệu cổng socket mà mình sử dụng. Mỗi cổng giao tiếp thể hiện một địa chỉ xác định trong hệ thống.

Số hiệu cổng gán cho socket phải duy nhất trên phạm vi máy tính đó, có giá trị trong khoảng từ 0 đến 65535 (16 bit). Trong thực tế thì các số hiệu cổng từ 0 đến 1023 (gồm có 1024 cổng) đã dành cho các dịch vụ nổi tiếng như: http: 80, telnet: 21, ftp: 23, ... Nếu chúng ta không phải là người quản trị thì nên dùng từ cổng 1024 trở lên.

3. Mô hình Client/Server.

Mô hình client/server sử dụng socket ở chế độ hướng kết nối TCP:



Hình 7 : Mô hình client-server sử dụng socket ở chế độ kết nối TCP

- Giai đoạn 1: Server tạo socket, gán số hiệu cổng và lắng nghe yêu cầu kết nối.
 - *socket()*: Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
 - *bind()*: Server yêu cầu gán số hiệu cổng (port) cho socket.
 - *listen()*: Server lắng nghe các yêu cầu nối kết từ các client trên cổng đã được gán.
- Giai đoạn 2: Client tạo Socket, yêu cầu thiết lập một nối kết với Server.
 - *socket()*: Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho socket của Client.
 - *connect()*: Client gửi yêu cầu kết nối đến Server có địa chỉ IP và Port xác định.
 - *accept()*: Server chấp nhận kết nối của Client, khi đó một kênh giao tiếp ảo được hình thành, Client và Server có thể trao đổi thông tin với nhau thông qua kênh ảo này.
- Giai đoạn 3: Trao đổi thông tin giữa Client và Server

Sau khi chấp nhận yêu cầu kết nối, thông thường Server thực hiện lệnh *read()* và nghe cho đến khi có thông điệp yêu cầu (Request Message) từ Client gửi đến. Server phân tích và thực thi yêu cầu. Kết quả sẽ được gửi về client bằng lệnh *write()*.

Sau khi gửi yêu cầu bằng lệnh *write()*, client chờ nhận thông điệp kết quả (ReplyMessage) từ Server bằng lệnh *read()*.

➤ Giai đoạn 4: Kết thúc phiên làm việc.

Các câu lệnh *read()*, *write()* có thể được thực hiện nhiều lần (ký hiệu bằng hình ellipse). Kênh ảo sẽ bị xóa khi Server hoặc Client đóng socket bằng lệnh *close()*.

II. PHÂN TÍCH THIẾT KẾ HỆ THỐNG.

1. Phân tích yêu cầu.

1.1. Giới thiệu về cách chơi cờ Đam (checkers).

- **Bàn cờ :** cờ đăm chơi trên mặt bàn cờ 8x8 .
- **Quân cờ :** sẽ bao gồm 24 quân cờ chia làm 2 màu, mỗi bên 12 quân.
- **Cách sắp xếp cờ Đam :**

Đặt bàn cờ Đam ở giữa hai người chơi, đặt hướng bàn cờ sao cho ô cuối cùng bên phải của hàng gần với mỗi người chơi là ô màu Trắng (hoặc màu sáng hơn).

Mỗi người chơi cần đặt 12 các quân của mình vào 12 ô màu tối ở 3 hàng đầu tiên gần với mình. Mỗi hàng sẽ có tổng cộng 4 quân cờ. Hai hàng ở giữa để trống.

- **Di chuyển quân cờ:**

Người chơi quân đen sẽ đi đầu tiên. Các quân cờ Đam chỉ có thể di chuyển theo đường chéo về phía trước khi bắt đầu trò chơi nếu không bị chặn ở phía trước và phải ở trong các ô tối màu.

Đến lượt chơi của đối thủ, các quân cờ cũng phải di chuyển theo nguyên tắc này.

- **Nhảy và bắt quân đối phương:**

Nếu quân cờ của bạn ở gần với một quân cờ của đối phương theo đường chéo, thì bạn có thể nhảy và bắt quân của đối phương bằng cách di chuyển hai ô theo hướng có quân đối phương giống như nhảy qua.

Khi quân cờ đối phương bị bắt, bạn có thể bỏ chúng ra khỏi bàn cờ.

- **Một số lưu ý khi nhảy và bắt quân:**

Đó là ô mà bạn nhảy tới phải là một ô trống.

Nếu bạn có cơ hội nhảy qua và bắt đối phương, thì bạn buộc phải thực hiện điều đó.

Nếu bạn có nhiều phương án nhảy, bạn có thể chọn một trong số chúng để thực hiện.

Nếu vị trí bạn nhảy tới lại tạo cho bạn một cơ hội Nhảy và bắt quân thì bạn phải tiếp tục cho đến khi nào không còn bắt được quân nào nữa.

- **Phong cấp Vua cho quân cờ**

Khi quân cờ của bạn đi đến cuối bàn cờ của đối phương thì chúng sẽ được phong Vua. Để phong cấp, bạn có đặt một “vương miện” nào đó của riêng bạn lên quân cờ hoặc đơn giản là trồng thêm một quân cờ khác lên nhằm phân biệt với các quân cờ khác. Quân được phong Vua có thể tiến lên và lùi lại do đó nó sẽ dễ dàng Bắt quân hơn.

Các quân Vua này cũng chỉ di chuyển được 1 ô trên một đường chéo nhưng bù lại trong một lượt đi của mình chúng có thể lùi hoặc tiến liên tục để Bắt quân.

Một số loại cờ Đam có hình vương miện ở mặt sau, do đó bạn chỉ cần lật quân cờ lên là được. Và bạn không bị giới hạn số Vua được phong.

- **Kết thúc ván chơi.**

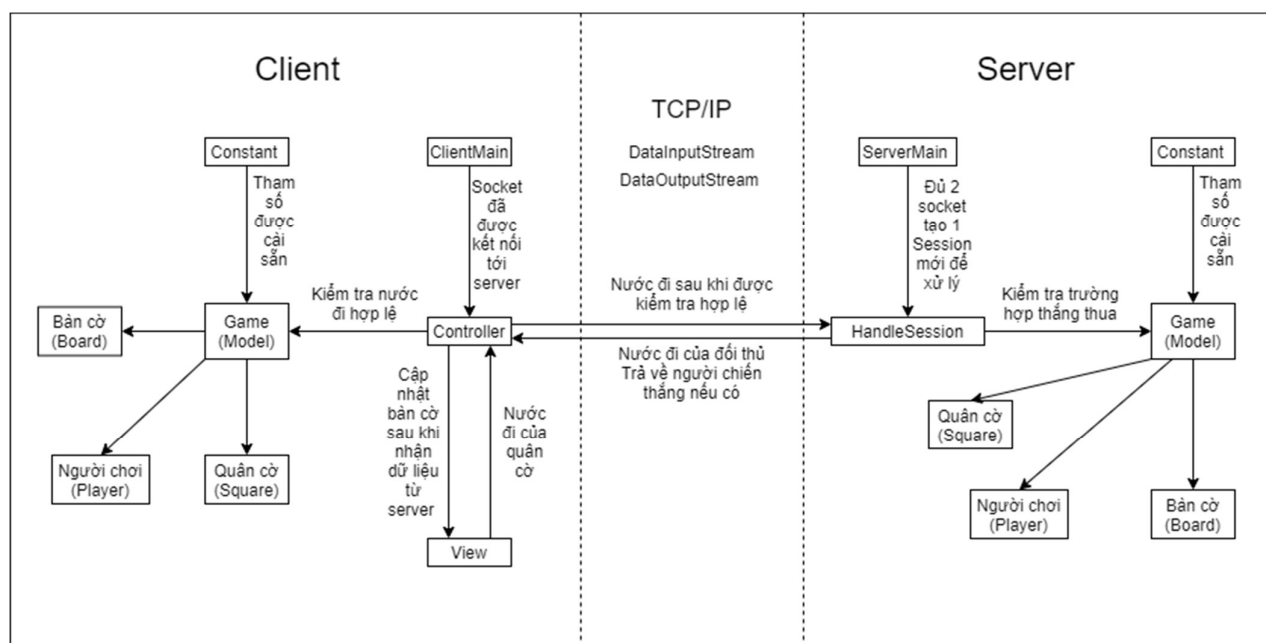
Tiếp tục các lượt chơi cho đến khi tất cả quân cờ của một bên bị bỏ hết ra khỏi bàn cờ. Khi đó, người còn quân cờ trên bàn duy nhất là người thắng cuộc.

1.2. Giới thiệu về đánh cờ Đam (checkers) qua mạng LAN.

Đánh cờ Đam (checkers) qua mạng LAN tức là người chơi sử dụng một chương trình có cài đặt thuật toán cờ Đam như đề cập ở phần giới thiệu trên để chơi với nhau thông qua môi trường mạng.

2. Phân tích hệ thống.

2.1. Sơ đồ hệ thống chương trình cờ Đam (checkers).



Hình 8: Sơ đồ hệ thống chương trình cờ Đam (checkers)

2.2. Phân tích hệ thống.

2.2.1. Giải thích các thành phần hệ thống.

- Server gồm các thành phần :

- ServerMain: Tạo Server tại cổng có số hiệu được đọc từ file config.properties, nhận 2 socket mới từ Client rồi gửi cho HandleSession để xử lý.
- HandleSession: Nhận 2 socket từ ServerMain, nhận và gửi dữ liệu qua lại giữa các socket và Model.
- Game (Model): Xử lý dữ liệu từ HandleSession thông qua các Constant được cài sẵn và các Model con như: Quân cờ (Square), Bàn cờ (Board) và Người chơi (Player).
- Client gồm các thành phần :
 - ClientMain: Tạo socket kết nối đến Server có địa chỉ IP và cổng port được đọc từ file config.properties, gửi socket cho Controller để xử lý.
 - Controller: Nhận socket từ ClientMain để giao tiếp với Server, giao tiếp với View và Game(Model).
 - View: Chứa các hàm xây dựng giao diện người dùng, lấy giá trị từ giao diện để tương tác với Controller.
 - Game (Model): Xử lý dữ liệu từ Controller thông qua các Constant được cài sẵn và các Model con như: Quân cờ (Square), Bàn cờ (Board) và Người chơi (Player).

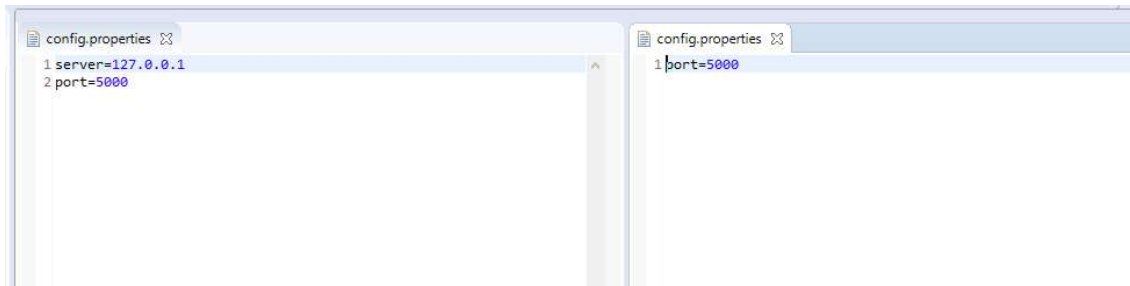
2.2.2. Giải thích mô hình sơ đồ hệ thống.

- ServerMain sẽ tạo Server tại cổng được đọc từ file config.properties, sau khi Server được tạo sẽ đợi kết nối socket từ Client.
- ClientMain sẽ tạo socket kết nối đến Server thông qua địa chỉ IP và cổng port được đọc từ file config.properties, sau khi tạo socket thành công sẽ truyền socket đến Controller nhằm mục đích sử dụng để truyền thông tin qua lại giữa Server và Client qua giao thức TCP/IP.
- Khi Server có đủ 2 socket kết nối tới sẽ chia thành 1 session riêng, HandleSession sẽ áp dụng đa luồng để xử lý riêng từng session chứa 2 socket, có thể xem như 1 phòng riêng dành cho 2 người chơi.
- Người chơi sẽ tương tác với hệ thống thông qua giao diện được cài đặt qua View, dữ liệu các nước đi hiện tại sẽ được gửi đến Controller.
- Controller sẽ dùng các Model chính như Game để kiểm tra nước đi hợp lệ nếu hợp lệ sẽ thể hiện lên View đồng thời gửi cho Server thông qua socket.
- Server khi nhận được dữ liệu từ người chơi sẽ cập nhật lên Model riêng để kiểm tra thắng thua đồng thời gửi dữ liệu đã qua xử lý cho cả hai người.
- Người chơi nếu nhận được dữ liệu là lượt mình thì được đi, không phải lượt thì đợi đến phiên. Nếu có người thắng thì sẽ được thông báo và trò chơi kết thúc.

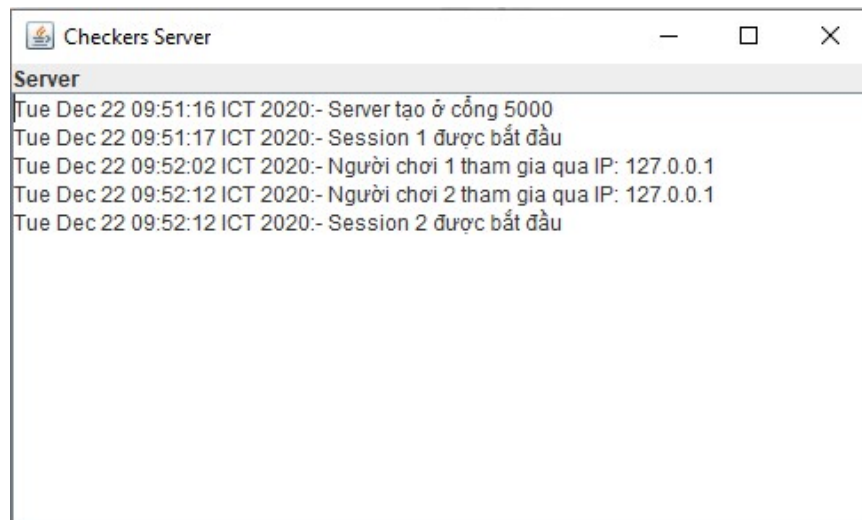
2.2.3. Giải thích cách thức hoạt động của xử lý đa luồng.

- Hàm `HandleSession` được thiết lập đa luồng để có thể xử lý được từng phòng dành cho 2 người chơi riêng biệt.
- Khi Server nhận đủ socket của 2 Client riêng biệt sẽ gọi đến hàm `HandleSession` từ đó tạo ra luồng riêng tương ứng với 2 người chơi.
- Từng luồng riêng biệt sẽ lần lượt nhận dữ liệu từ từng người chơi, cập nhật lên Model riêng để xử lý thắng thua rồi gửi trả về phía Client.

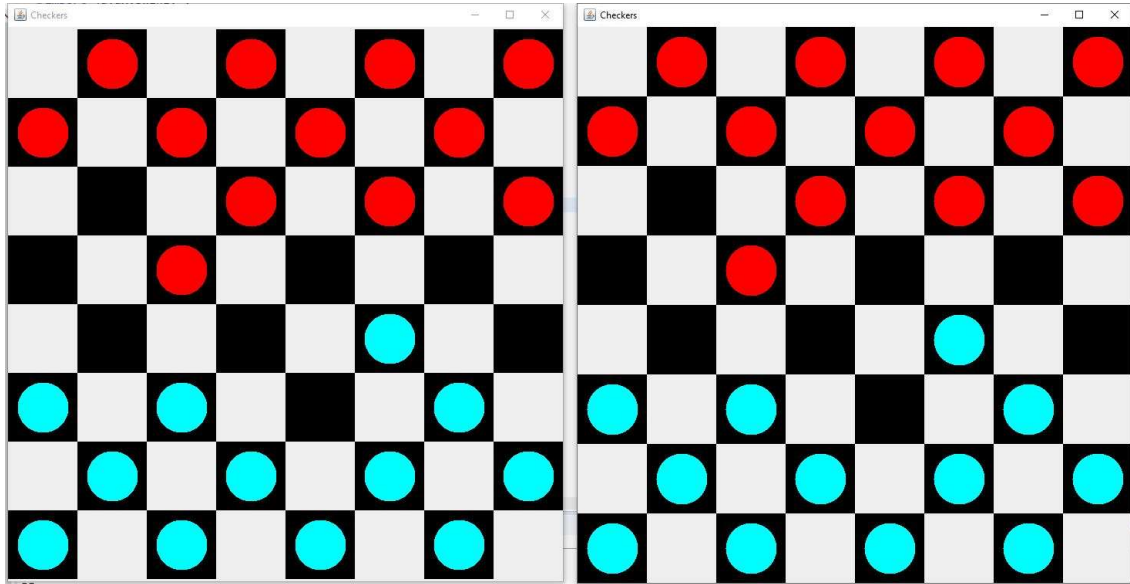
3. Kết quả.



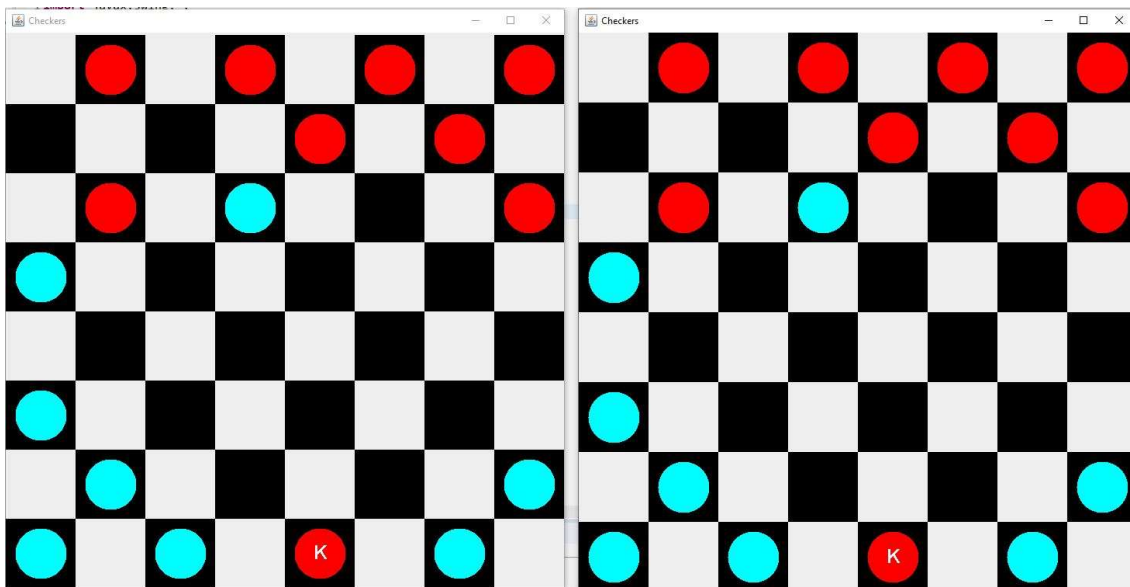
Hình 9: Địa chỉ IP của Server và cổng port tương ứng được nhập vào 2 file `config.properties` (Client phía bên trái, Server phía bên phải)



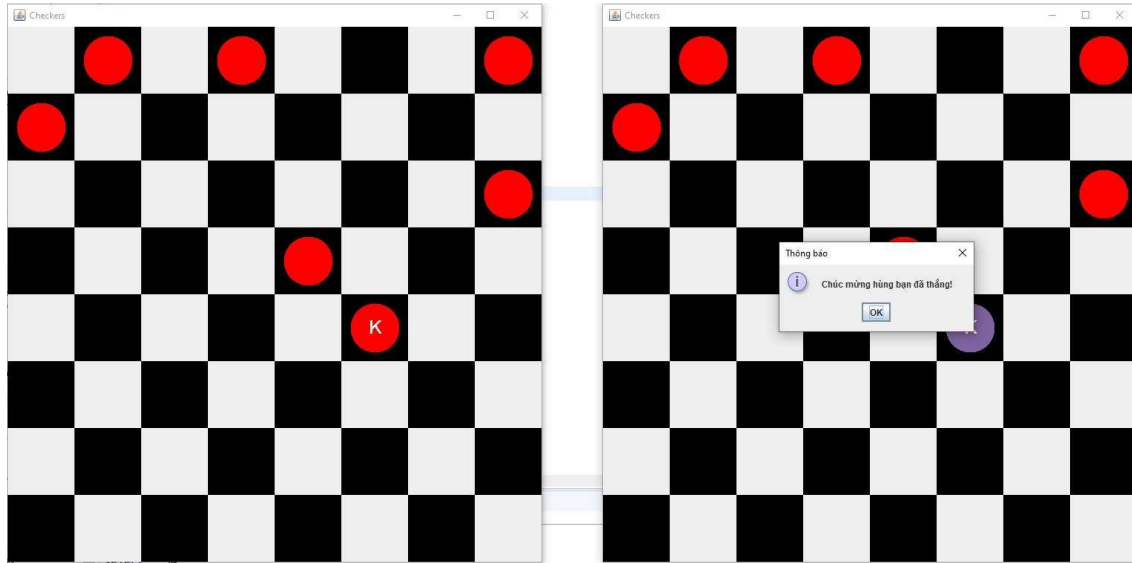
Hình 10: Giao diện của Server



Hình 11: Giao diện của Client (đang thử nghiệm trên cùng 1 máy)



Hình 12: Trường hợp quân đỏ phong vua khi có quân đi hết bàn cờ (quân K)



Hình 13: Trường hợp quân đỏ thắng

- Server có thể xử lý nhiều phòng (2 client) cùng một lúc nhờ được cài đặt đa luồng.
- Chương trình phản hồi nhanh.

III. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

- Thông qua đồ án nhóm em có thể hiểu sâu hơn về môn học lập trình mạng.
- Hiểu sâu về mô hình Client / Server.
- Hiểu về mô hình mạng 7 tầng.
- Vận dụng tốt ngôn ngữ lập trình Java và Thread trong Java.
- Hiểu sâu về lập trình với Socket trong Java.

2. Hướng phát triển

- Nghiên cứu thêm về thiết kế giao diện giúp ứng dụng đẹp hơn.
- Có thể mở rộng một số chức năng như lưu danh sách người thắng, tái đấu hay thêm đồng hồ bấm giờ giúp tăng thêm kịch tính.
- Ứng dụng JSP/Servlet để có thể đưa trò chơi lên môi trường web giúp thuận tiện cho việc tham gia thi đấu.

TÀI LIỆU THAM KHẢO

I. Hệ điều hành

Bài giảng hệ điều hành:

http://monhoc.vn/tai-lieu/bai-giang-he-dieu-hanh-chuong-6-deadlocks-3368/?fbclid=IwAR26grSKun3Cp7g_ipTMLN0gHV6yLzD2YxpSXY316E8GEhQfwTeVmCWOAdU

Giáo trình hệ điều hành đại học Cần Thơ:

<http://monhoc.vn/tai-lieu/giao-trinh-he-dieu-hanh-deadlock-1253/>

II. Lập trình mạng

TCP/IP - Bách khoa toàn thư mở Wikipedia:

<https://vi.wikipedia.org/wiki/TCP/IP>

Hướng dẫn chơi cờ đam – thuthuatchoi.com:

<https://thuthuatchoi.com/huong-dan-cach-choi-co-dam-checker.html>

Slides Giáo trình giảng dạy môn Lập trình mạng – Giảng viên Phạm Minh Tuấn - Mai Văn Hà