





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



BÁO CÁO ĐỒ ÁN
CƠ SỞ NGÀNH MẠNG

Đà Nẵng, 12/2018

Đà Nẵng, 12/2018



Nhận xét của giáo viên hướng dẫn

Nhận xét:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm (bằng số) :

Điểm (bằng chữ):.....

Ký xác nhận

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, em đã nhận được rất nhiều sự quan tâm 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 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 Nguyễn Thị Lệ Quyên đã tận tình giúp đỡ 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, em không thể tránh được sai sót. Vậy nên, em rất mong quý thầy/cô thông cảm cho những sai sót ấy và ghi nhận những gì em đã làm được.

Một lần nữa, 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, 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 2018

Sinh viên thực hiện

VÕ VĂN TRINH

Contents

Tóm tắt đồ án	4
PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH:	4
❖ 5PHẦN II: LẬP TRÌNH MẠNG :	4
❖ 5	
PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH	4
I. 5	
1. 5	
Yêu cầu của đề tài	4
II. 6	
1. 6	
2. 7	
III. 8	
1. 8	
2. 9	
3. 9	
IV. 11	
PHẦN II: LẬP TRÌNH MẠNG	10
TIÊU ĐỀ: Xây dựng chương trình đánh bài cào qua mạng LAN	10
I. 12	
1. 12	
2. 13	
3. 14	
II. 16	
1. 16	
2. 16	
3. 19	
III. 19	
1. Kết luận	18
2. Hướng phát triển	18

Tóm tắt đồ án

Đồ án Cơ sở ngành Mạng của em gồm 2 phần:

PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH:

- ❖ Chương trình mô phỏng giải thuật nhà băng của Dijkstra để tránh deadlock

PHẦN II: LẬP TRÌNH MẠNG :

- ❖ Xây dựng chương trình đánh bài cào qua mạng LAN

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 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:

- 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
- Chúng ta có thể cho phép hệ thống đi vào trạng thái deadlock, phát hiện nó và phục hồi.
- 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 $\text{Need}[i, j] = k$, 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, $\text{Need}[i, j] = \text{Max}[i, j] - \text{Allocation}[i, j]$.

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 $\text{Work} := \text{Available}$ và $\text{Finish}[i] := \text{false}$ cho $i = 1, 2, \dots, n$.

Bước 2: Tìm i thỏa:

a) $\text{Finish}[i] = \text{false}$

b) $\text{Need}[i] \leq \text{Work}$. Nếu không có i nào thỏa, di chuyển tới bước 4

Bước 3: $\text{Work} := \text{Work} + \text{Allocation}[i]$ và $\text{Finish}[i] := \text{true}$ Di chuyển về bước 2.

Bước 4: Nếu $\text{Finish}[i] = \text{true}$ 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.

2.4 Thuật toán yêu cầu tài nguyên

- Cho Requesti là vector yêu cầu cho quá trình P_i . Nếu $\text{Requesti}[j] = k$, thì quá trình P_i muốn k thể hiện của loại tài nguyên R_j . Khi một yêu cầu tài nguyên được thực hiện bởi quá trình P_i , thì các hoạt động sau được thực hiện:

Bước 1: Nếu $\text{Requesti} \leq \text{Needi}$, di chuyển tới bước 2. Ngược lại, phát sinh một điều kiện lỗi vì quá trình vượt quá yêu cầu tối đa của nó.

Bước 2: Nếu $\text{Requesti} \leq \text{Available}$, di chuyển tới bước 3. Ngược lại, P_i phải chờ vì tài nguyên không sẵn có.

Bước 3: Giả sử hệ thống cấp phát các tài nguyên được yêu cầu tới quá trình P_i bằng cách thay đổi trạng thái sau:

$\text{Available} := \text{Available} - \text{Requesti};$

$\text{Allocationi} := \text{Allocationi} + \text{Requesti};$

$\text{Needi} := \text{Needi} - \text{Requesti};$

Nếu kết quả trạng thái cấp phát tài nguyên là an toàn, thì giao dịch được hoàn thành và quá trình P_i được cấp phát tài nguyên của nó. Tuy nhiên, nếu trạng thái mới là không an toàn, thì P_i phải chờ Requesti và trạng thái cấp phát tài nguyên cũ được phục hồi.

III. THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH

1. Phân tích yêu cầu

Bài toán đặt ra.

Số lượng các nguồn tài nguyên, các yêu cầu của hệ thống được tạo ngẫu nhiên

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.

- p: Số tiến trình
- r: Số tài nguyên

2. Xây dựng chương trình

2.1 Các biến, hàm sử dụng trong chương trình

STT	Tên hàm, biến	Chức năng
1	Alloc[][]	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	R	Số tài nguyên
6	P	Số tiến trình
7	CreateData()	Khởi tạo dữ liệu ban đầu
8	PrintData()	Hiển thị dữ liệu khởi tạo ra màn hình Console
9	Update_Q(int k)	Cập nhật lại tài nguyên sau khi 1 tiến trình hoàn thành
10	RunProcessor()	Khởi chạy hệ thống

3. Kết quả thử nghiệm.

- Hệ thống an toàn :



Hình 2 : Kết quả kiểm tra hệ thống an toàn

- Hệ thống không an toàn:

ĐỒ ÁN HỆ ĐIỀU HÀNH

Xây Dựng Chương Trình Mô Phỏng Giải Thuật Nhà Bảng Của Dijkstra Để Tránh Deadlock

SỐ TIẾN TRÌNH P: SỐ TÀI NGUYÊN R:

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)	5	8	7	9	0	3	5	1	8	
P(2)	8	8	5	3	7	5	4	2	8	
P(3)	7	3	4	3	4	1	2	5	8	
P(4)	8	7	9	2	4	0	1	8	5	
P(5)	8	8	8	0	8	8	5	2	7	
P(6)	8	8	2	4	7	7	4	8	2	
P(7)	8	8	8	7	2	3	5	2	8	
P(8)	3	8	4	4	2	8	2	8	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)	8	8	3	3	0	3	2	0	8	
P(2)	8	4	5	8	5	3	3	0	5	
P(3)	3	0	1	2	2	1	2	3	8	
P(4)	5	8	8	1	3	4	1	2	3	
P(5)	2	8	8	8	4	4	8	2	4	
P(6)	3	4	1	1	7	5	3	8	1	
P(7)	0	8	5	8	2	2	5	0	0	
P(8)	3	8	8	8	3	2	0	0	7	

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)	3	1	4	6	0	0	4	1	2	
P(2)	5	4	0	3	2	2	1	3	4	
P(3)	2	3	3	8	2	0	8	2	0	
P(4)	4	1	0	1	4	5	8	7	2	
P(5)	8	8	0	8	5	4	1	1	3	
P(6)	5	2	1	5	0	2	1	9	2	
P(7)	8	0	1	7	5	1	1	3	5	
P(8)	8	0	4	4	2	8	3	8	1	

BẢNG DỮ LIỆU AVAILABLE

	R(1)	R(2)	R(3)	R(4)	R(5)	R(6)	R(7)	R(8)	R(9)	R(10)
	3	0	5	2	0	8	3	8	8	

KHOẢNG CHẠY & KIỂM TRA

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



Hình 4 : Kết quả kiểm tra hệ thống không an toàn

IV. ĐÁNH GIÁ KẾT QUẢ

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.
5. Thực hiện đề tài mô phỏng thành công thuật toán Banker để tránh Deadlock, hiểu rõ hơn cách mà hệ điều hành quản lý phân 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.

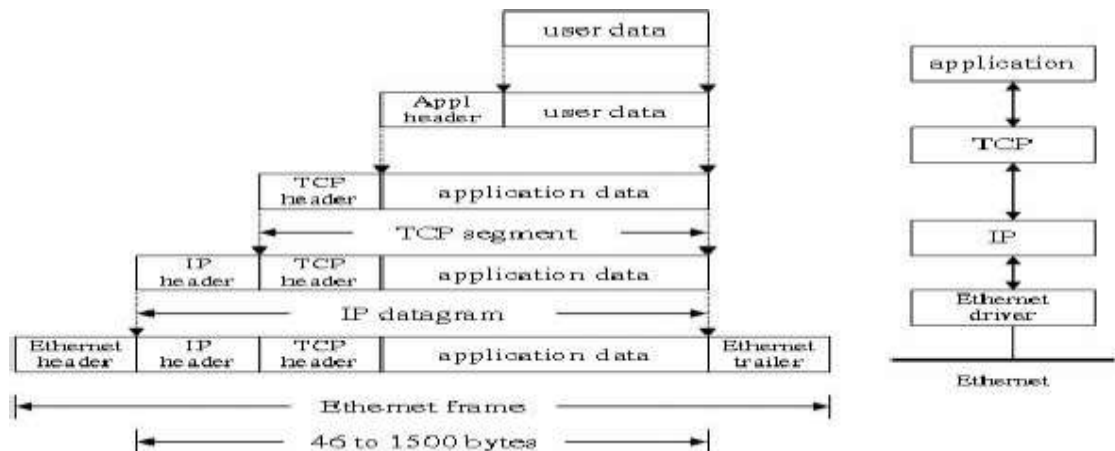
PHẦN II: LẬP TRÌNH MẠNG

TIÊU ĐỀ: Xây dựng chương trình đánh bài cào qua mạng LAN

I. CƠ SỞ LÝ THUYẾT

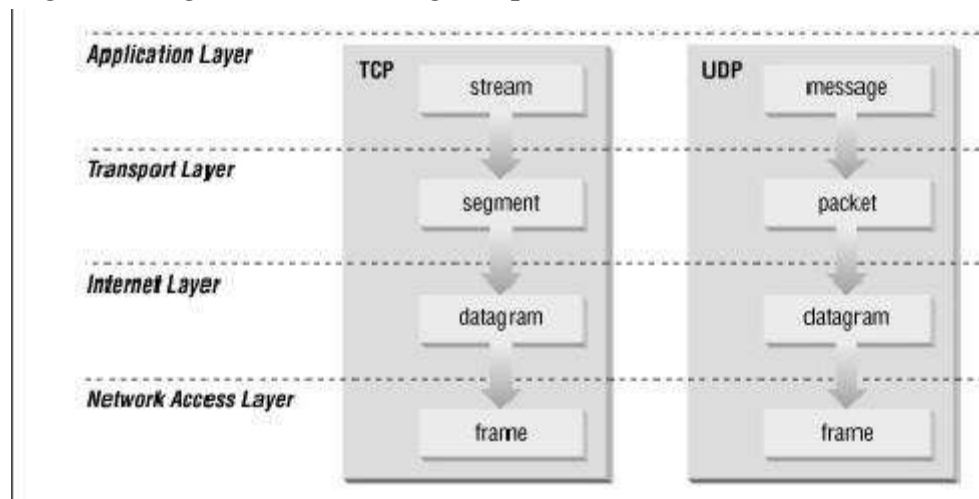
1. Giao thức TCP/IP

I.1. Phương thức hoạt động của bộ giao thức TCP/IP.



Hình 5: 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 6: 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.

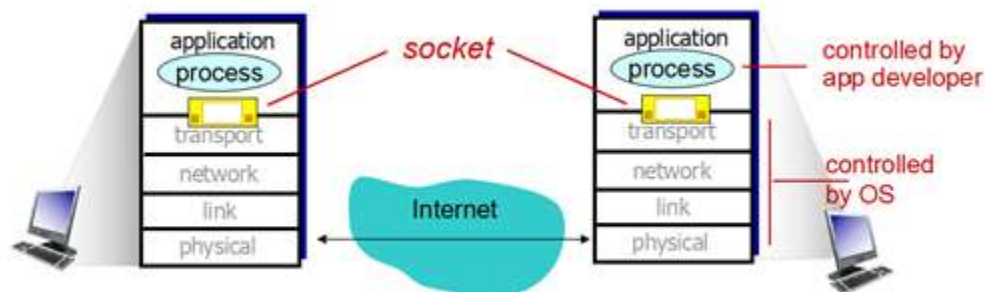
2.1. Khái niệm Socket và cổng port

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 7: Socket

2.1.2. Số hiệu cổng của Socket

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.

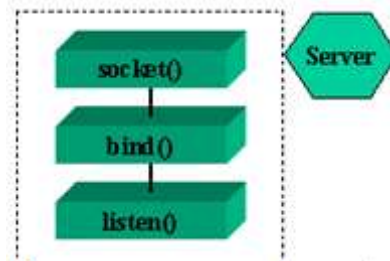
Số hiệu cổng gán cho socket phải duy nhất trên phạm vi máy tính đó, có giá trị trong khoảng từ 0 đến 65535 (16 bit). Trong thực tế thì các số hiệu cổng từ 0 đến

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

➤ Giai đoạn 1: Server tạo socket, gán số hiệu cổng và lắng nghe yêu cầu kết nối.



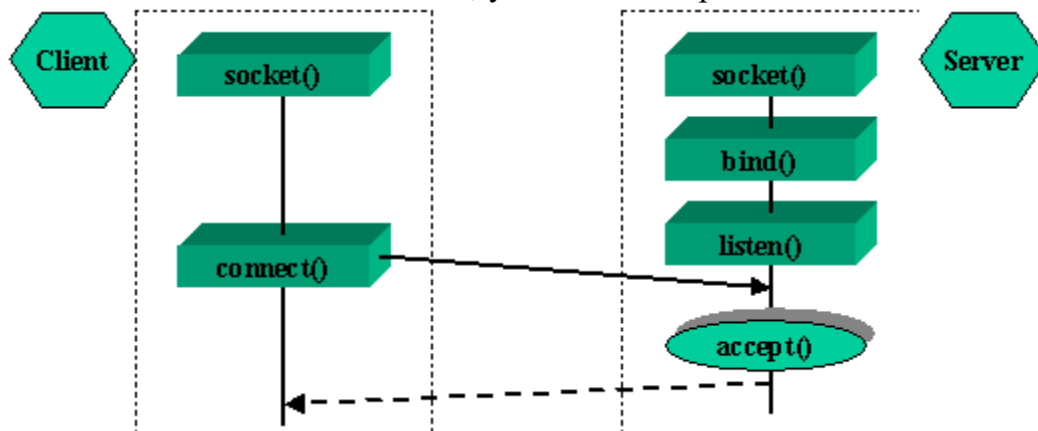
Hình 8 : Mô hình client-server

- *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.



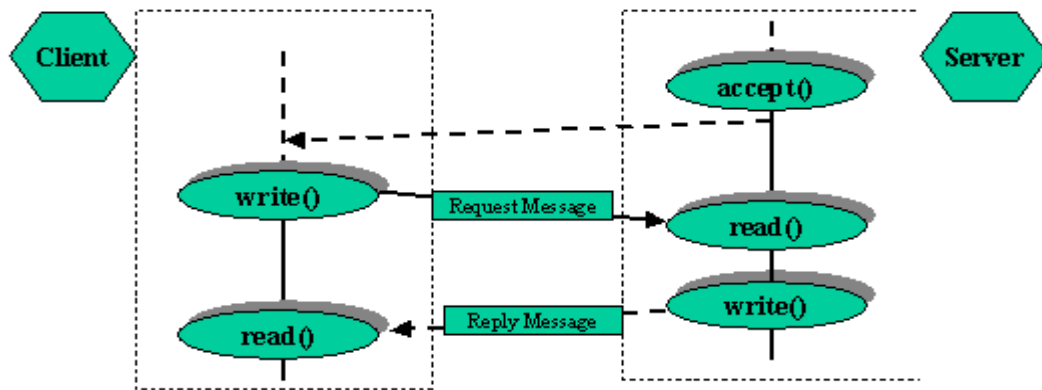
Hình 9: Socket giữa client-server

- *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



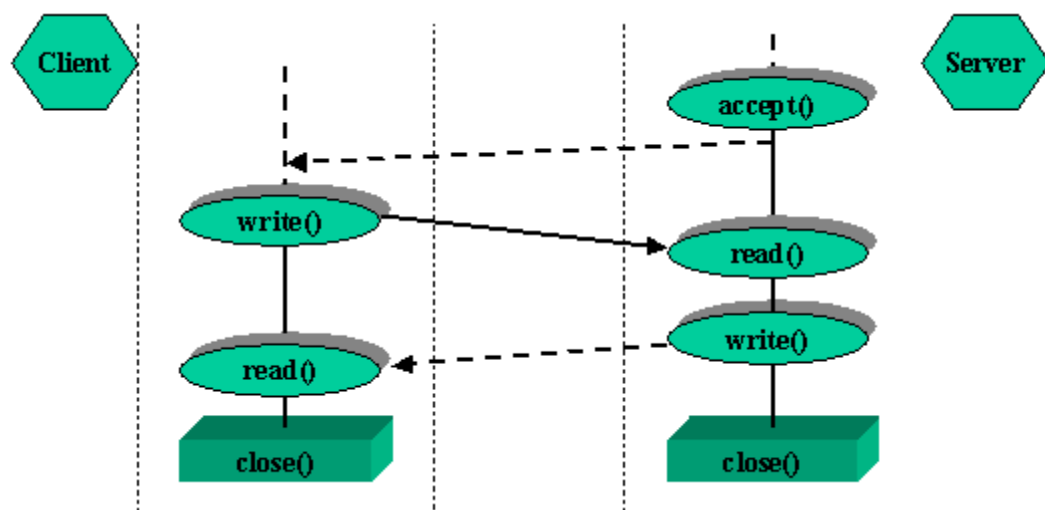
Hình 10: Trao đổi thông tin giữa Client và Server

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.



Hình 11: 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

❖ Giới thiệu về cách chơi đánh bài cào

- Trong bài cào số lượng người chơi từ 2 đến 5 người trong một lần đánh. Mỗi người được chia 3 lá bài, sau đó chia thành từng đôi so bài với nhau. Căn cứ vào loại bài để phân định thắng thua.

- Chơi bài cào sử dụng 52 lá bài trong bộ bài Joker. Trong đó những cây từ A đến 9 cũng chính là số điểm từ 1 đến 9, lá 10 là 0 điểm các lá J, Q, K là bài công. Sau khi nhận 3 lá bài sẽ so sánh 3 lá bài mình với người khác. Mỗi lá bài đều có qui định tính điểm riêng, vì thế căn cứ vào loại lá bài để tính điểm.

- Quy tắc tính điểm cơ bản là J, Q, K là bài công, không tính điểm. Lá bài 10 không tính điểm nhưng không được gọi là bài công. Lá bài từ A đến 9 thì được tính điểm. Cộng điểm của 3 lá bài lại, dùng điểm đó để phân định thắng thua.

Việc phân định thắng thua chủ yếu dựa vào tổng điểm các lá bài từ A đến 9 và tổ hợp bài công.

- Quy tắc so bài : Bài công > bài điểm ($K > Q > J > 10 > 9 > \dots > 2 > A$). Ngoài ra còn một vài quy tắc chơi khác. Nếu điểm bằng nhau thì có thể so tổ hợp các bài công (vd : $KQ9 > QJ9$).

- Quy tắc thắng thua: sau khi so bài mình với người khác, người thắng là người có điểm số cao hơn. Lấy hệ số cuối cùng để phân định thắng thua.

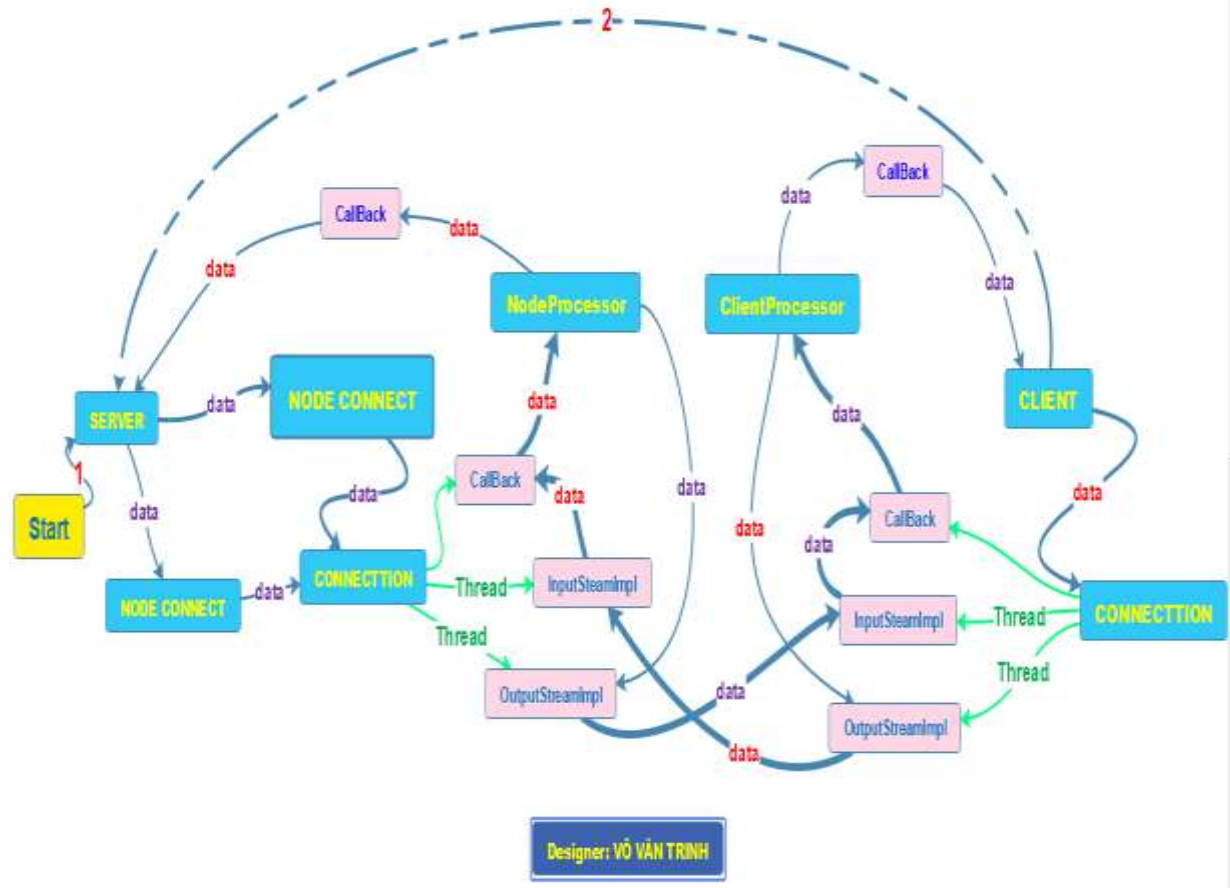
❖ Giới thiệu về đánh bài cào qua mạng LAN

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

2. Phân tích hệ thống

❖ Sơ đồ hệ thống chương trình đánh bài cào

- Sơ đồ hệ thống



Hình 12: Sơ đồ hệ thống đánh bài cào qua mạng LAN

❖ Phân tích hệ thống

- Giải thích các thành phần trong hệ thống :

- Server gồm các thành phần :
 - Server : là 1 class, chấp nhận các kết nối từ client và tạo NodeConnect
 - NodeConnect : là 1 class gồm các method gửi và nhận dữ liệu từ client tới thông qua Connection do có hai luồng đang chạy là OutputStream và InputStream.
 - NodeProcessor : là class xử lý dữ liệu từ client gửi lên có 2 trường hợp xảy ra. Một là xử lý xong gửi về Client thông qua luồng OutputStream hoặc hai là gửi lên server thông qua Method Callback() mang dữ liệu rồi Server, lúc này Server xử lý và gửi lại tất cả các Client.
- Client gồm các thành phần :
 - Client: là 1 class, tạo ra 1 socket để kết nối tới Server đồng thời là GUI của chương trình.

- Connection : là 1 class, bao gồm 2 luồng InputStream để nhận dữ liệu từ Server gửi về và một luồng OutputStream để gửi dữ liệu đến các NodeConnect.
- ClientProcessor : là class xử lý dữ liệu khi Server gửi về có 2 trường hợp xảy ra. Một là xử lý xong thông qua luồng OutputStream gửi dữ liệu về lại Server, 2 là thông qua Method Callback() gửi dữ liệu về Client (GUI) để client xử lý rồi tiếp tục gửi lên Server.

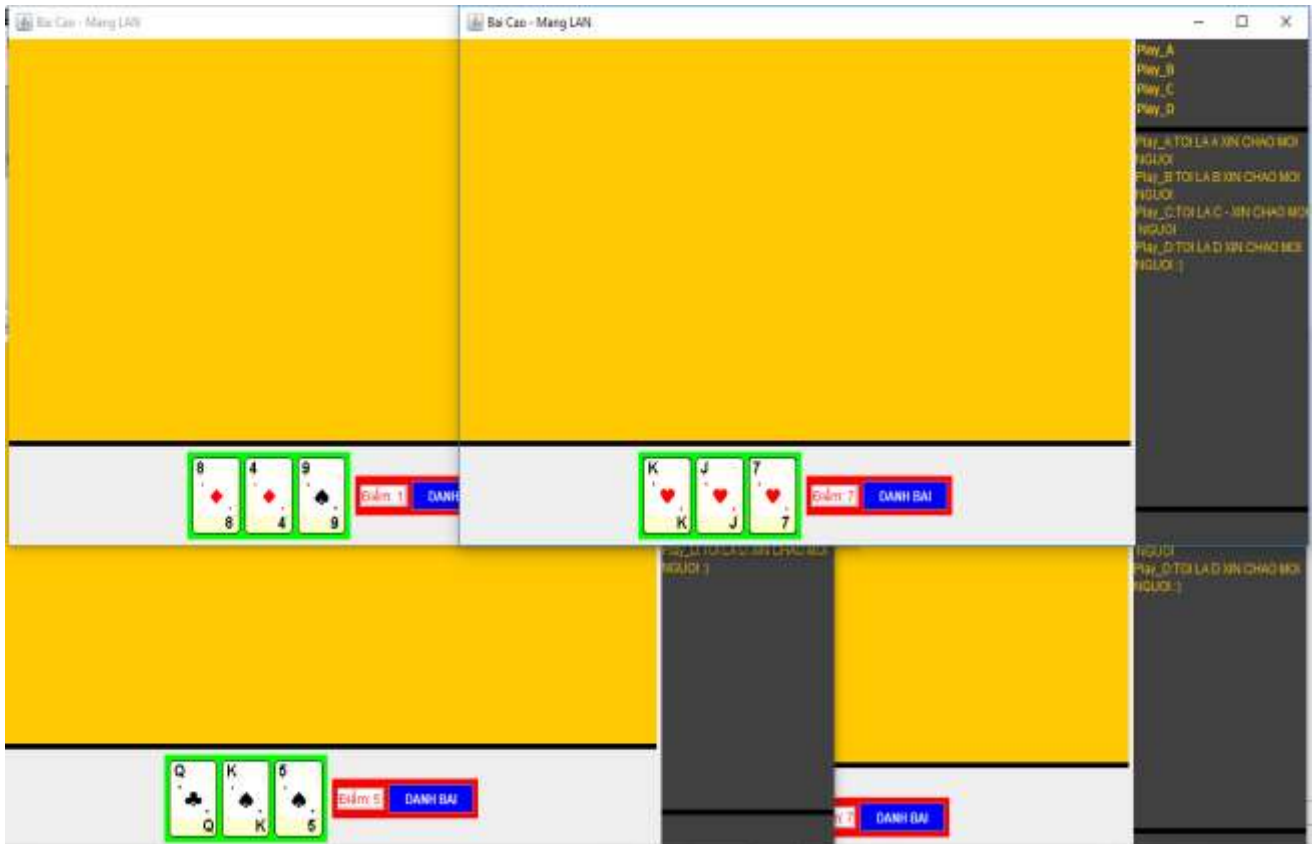
- Giải thích mô hình sơ đồ hệ thống :

- + Bước 1 : Server khởi chạy tạo ServerSocket và gán cổng Port = 12345, bắt đầu lắng nghe, đợi Client kết nối tới.
- + Bước 2 : Khi các client tạo 1 Socket với cổng 12345, kết nối tới Server. Lúc này Server sẽ chấp nhận kết nối đồng thời mỗi Client kết nối tới sẽ tạo ra 1 NodeConnect. NodeConnect được coi như là thẳng trung gian trao đổi dữ liệu giữa Client và Server.
- + Bước 3: Trao đổi dữ liệu trong hệ thống khi đã có các kết nối được chấp nhận :
 - Khi mỗi NodeContion được tạo ra từ mỗi Client thì NodeContion sẽ gửi 1 key 6 kí tự (tất các Key đều 6 kí tự) “**S_Nhan**” về client thông qua luồng OutputStream, lúc này luồng InputStream sẽ nhận và thông qua Method Callback() để chuyển key về ClientProcessor lúc này Client nhận Key để xác nhận đã kết nối tới Server.
 - Tiếp tục Client lại gửi lên NodeConnection Key “**C_Nhan**” thông qua luồng chạy, NodeProcessor nhận Key và gửi Key “**S_Name**” để lấy thông tin Tên người chơi, tại ClientProcessor sẽ chuyển Key sang GUI để lấy tên của người chơi.
 - Lúc này tại Client (GUI) sẽ gửi mã “C_Name” lên NodeProcessor lúc này NodeProcessor sẽ không gửi lại Client mà gửi về Server thông qua Method Callback(), lúc này Server sẽ tiến hành tạo mảng các NodeConnection đồng thời nếu Size() của mảng các NodeConnection tức là số Client kết nối tới mà bằng 4 thì tiến hành gửi lại mỗi NodeConnection 1 Key gồm “3_Card”+strCLst. Trong đó strCLst là chuỗi được lấy từ Class Logic về việc chia bài. Key này được gửi tới mỗi Client và Client phân tích chuỗi strCLst từ Class Logic để đổ lên giao diện mỗi màn hình Client thành tổ hợp 3 lá bài bất kì.
 - Khi thực hiện xong vòng lặp chia bài, thì Server tiếp tục thực hiện thêm 1 vòng lặp đến tất các Client lần nữa lúc này Server sẽ gửi Key chứa điểm của mỗi Client và Show lên mỗi màn hình Client.

- Phân tích mô hình Chat giữa các Client

- + Bước 1 : Client (GUI) sẽ gửi dữ liệu gồm Key “C_CHAT”+nội dung tin nhắn thông qua luồng OutputStream.
- + Bước 2 : Luồng InputStream của chính Client sẽ đưa dữ liệu lên NodeProcessor của chính Client đó thông qua Method Callback().
- + Bước 3 : NodeProcessor sẽ tiếp tục thông qua Method Callback() gửi dữ liệu lên Server lúc này Server phân tích nhận được Key “C_CHAT” lập tức thực hiện vòng lặp gửi lại các NodeConnection và dữ liệu theo đường truyền gửi về Client và đổ lên GUI.

3. Kết quả



Hình 13: Kết quả chương trình đánh bài cào

III. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

- Thông qua đồ án 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 chuyên nghiệp.
- Hiểu sâu về lập trình với Socket trong Java

2. Hướng phát triển

- Xây dựng giao diện thân thiện với người chơi.

- Sử lý tốt về đồng bộ luồng trong hệ thống
- Hoàn thành về xử lý thắng cho người chơi

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-DAI-HOC-CAN-THO-3266/?FBCLID=IWAR0QQ-RXV0QOUHE2IBSEXLY2AKYNOUVSCNXGZIIKKGKU37K0UHFH1SW200Y0>

II. Lập trình mạng

Java Cryptography Architecture Standard Algorithm Name Documentation for
JDK 8 - Cipher (Encryption) Algorithms:

<https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html#Cipher>

SSH - Bách khoa toàn thư mở Wikipedia:

TCP/IP - Bách khoa toàn thư mở Wikipedia:

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

Slides lập trình Socket với TCP – Thầy Mai Văn Hà

--HẾT--