Chương 2
Tổng quát về lập trình mạng

Giảng viên: Nguyễn Đức Thái
thai@cse.hcmut.edu.vn

Nền tảng trong bài giảng trước

- Mô hình tham khảo OSI
- Tầng vận chuyển và các giao thức TCP, UDP
- Mô hình Client/Server
- Địa chỉ IP, địa chỉ port
- …

Nội dung

- Giới thiệu lập trình mạng
- Môi trường lập trình mạng
- Giới thiệu một số hàm được sử dụng trong lập trình mạng
- Lập trình sockets với TCP và UDP

Khái niệm lập trình mạng

- Lập trình mạng được hiểu là viết code chương trình giao tiếp với chương trình khác trong mạng máy tính
- Môi trường lập trình mạng:
  - Lập trình mạng trong Unix
  - Lập trình mạng trong Windows

Giao tiếp giữa các process trong mạng

- Chúng ta đề cập đến giao tiếp trong mô hình Client/Server
- Đầu tiên chương trình (hoặc process) tại phía server sẽ khởi động và chờ đợi kết nối từ chương trình (hoặc process) từ phía client
- Chương trình (hoặc process) phía client chạy và kết nối với chương trình (hoặc process) từ phía server
- Những process của server và client sẽ cùng nhau thiết lập một hệ phân bố (distributed system).
- Sự giao tiếp giữa process của server và process của client có thể
  - theo hướng kết nối (connection-oriented)
  - hoặc không kết nối (connectionless)

Internet socket

- *Internet socket* hoặc *network socket* là một điểm cuối (endpoint) trong giao tiếp hai chiều giữa những process trong mạng máy tính trên nền tảng IP, chẳng hạn như Internet

Giới thiệu lập trình với sockets

- Socket thường được hiện thực bởi một thư viện API (Application Programming Interface), như Berkeley sockets trong Unix.
- Những hàm tiêu biểu được cung cấp bởi thư viện API:

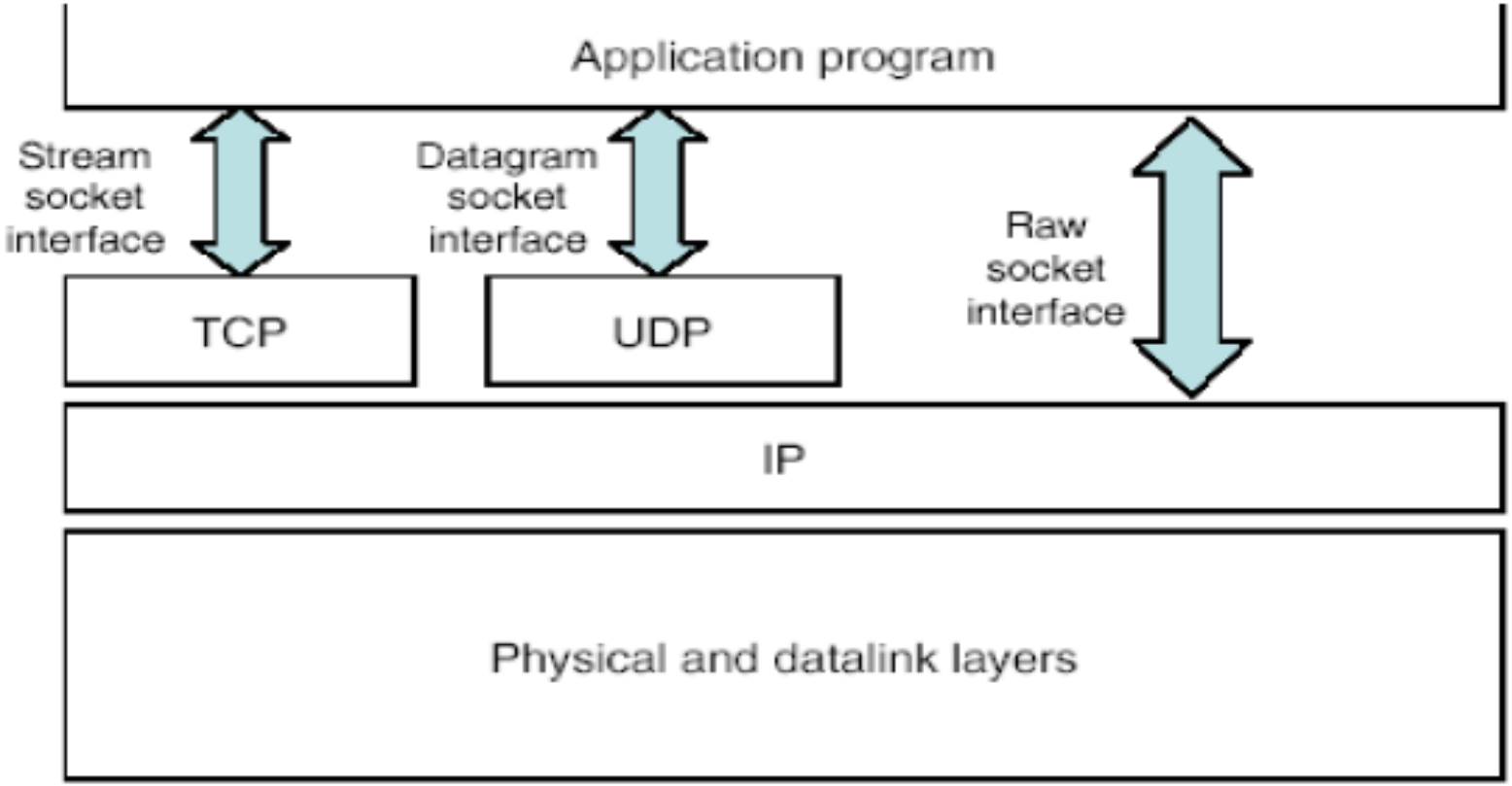
Primitives	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Lập trình mạng trong Unix

Các loại socket

- Các loại socket:
  - Giao tiếp socket định nghĩa 3 loại socket có thể dùng trên môi trường TCP/IP
    - **Stream Socket**: dùng cho giao thức hướng kết nối (connection-oriented protocol), chẳng hạn như TCP
    - **Datagram Socket**: dùng cho giao thức không kết nối (connectionless protocol), chẳng hạn như UDP
    - **Raw Socket**: dùng cho một số giao thức của các ứng dụng đặc biệt, dùng các dịch vụ trực tiếp của tầng IP

Lập trình socket trên Unix

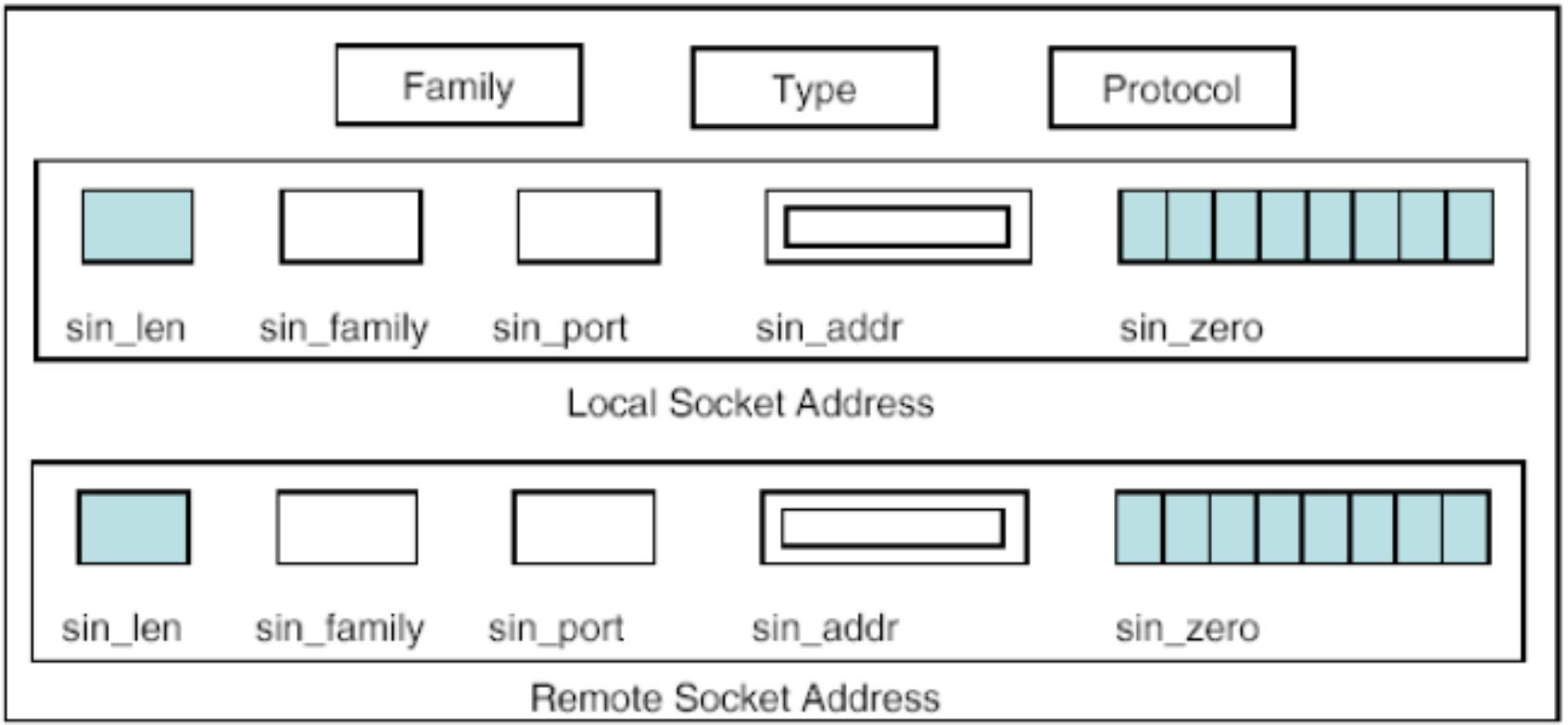


Hình - Các loại socket

Cấu trúc socket (1/2)

- Cấu trúc socket:
  - Socket được định nghĩa trong hệ điều hành bằng một cấu trúc, được xem như điểm nối để hai process giao tiếp với nhau
  - Cấu trúc socket gồm 5 trường (field) được mô tả như sau:
    - Family: xác định protocol group
    - Type: xác định loại socket (**stream**, **datagram** hay **raw** socket.
    - Protocol: là trường thường được gán giá trị mặc định bằng 0
    - Local Socket Address và Remote Socket Address: là địa chỉ socket của process cục bộ và từ xa

Cấu trúc socket (2/2)



Socket

Cấu trúc địa chỉ socket (1/5)

sockaddr_in

Hình - Cấu trúc địa chỉ socket

```
struct   sockaddr_in
{
        u_char              sin_len;
        u_short             sin_family;
        u_short             sin_port;
        struct    in_addr   sin_addr;
        char                sin_zero[8];
};
```

Cấu trúc địa chỉ socket (2/5)

- Cấu trúc địa chỉ socket trong IPv4, được gọi là sockaddr_in và được định nghĩa trong header <netinet/in.h>

```
struct in_addr {
        in_addr_t       s_addr;         /* 32-bit IPv4 address */
};                                      /* network byte ordered */


struct sockaddr_in {
        uint8_t         sin_len;        /* length of structure (16) */
        sa_family_t     sin_family;     /* AF_INET */
        in_port_t       sin_port;       /* 16-bit TCP or UDP port number */
                                        /* network byte ordered */
        struct in_addr  sin_addr;       /* 32-bit IPv4 address */
                                        /* network byte ordered */
        char            sin_zero[8];    /* unused */
};
```

Cấu trúc địa chỉ socket (3/5)

- Cấu trúc địa chỉ socket:
  - Địa chỉ này lưu trữ địa chỉ IP, chỉ số port, và dạng (family protocol)
  - Tên cấu trúc là sockaddr_in
    - sin_len: lưu trữ chiều dài cấu trúc của sockaddr_in
    - sin_family: dạng protocol của socket
    - sin_port: chỉ số port
    - sin_addr: địa chỉ Internet của socket
    - sin_zero[8]: không dùng, giá trị bằng 0

Các hàm dùng cho lập trình socket (1)

- int **socket** (int domain, int type, int protocol);

Trong đó:

- `domain`: họ địa chỉ, thường sử dụng là AF_INET: địa chỉ Internet
- `type` : kiểu socket (SOCK_STREAM, SOCK_DGRAM)
- `protocol` : giao thức được dùng, giá trị mặc định là 0

- int **bind** (int sockfd, struct sockaddr *my_addr, int addrlen);

Trong đó:

- `sockfd`: là socket file descriptor, từ hàm `socket`
- `my_addr`: một pointer đến struct sockaddr
- `addrlen` = sizeof(struct sockaddr).

Các hàm dùng cho lập trình socket (2)

- int **connect** (int sockfd, struct sockaddr *serv_addr, int addrlen);

Trong đó:

- `sockfd` là socket file descriptor, trả từ hàm `socket`
- `serv_addr` là struct sockaddr chứa port và IP address đích
- `addrlen` = sizeof(struct sockaddr).

- int **listen** (int sockfd, int backlog);

Hàm này được gọi sau hàm **`socket`** và **`bind`**, luôn được gọi trước hàm **`accept`**

- `sockfd` là socket file descriptor, trả từ hàm `socket`
- `backlog` là số kết nối cho phép của hàng đợi. Các yêu cầu `connect` của đối tác sẽ được lưu trong *queue* cho tới khi được `accept`

Các hàm dùng cho lập trình socket (3)

- int **accept** (int sockfd, void *addr, int *addrlen);

Trong đó:

  - `sockfd` là socket file descriptor.
  - *`addr`* là pointer trỏ tới sockaddr_in. Xác định ai kết nối tới, kết nối từ port nào.
  - `addrlen` là biến int, độ dài sizeof (struct sockaddr_in)

- int **send** (int sockfd, const void *msg, int len, int flags);
- int **recv** (int sockfd, void *buf, int len, unsigned int flags);
- int **read** (int sockfd, const void *buf, int len);
- int **write** (int sockfd, const void *buf, int len);

Các hàm dùng cho lập trình socket (4)

- int **sendto** (int sockfd, const void *msg, int len, unsigned int flags, const struct sockaddr *to, int tolen);
  - `tolen` có giá trị bằng sizeof (struct sockaddr).
  - Dùng cho lập trình socket với UDP

- int **recvfrom** (int sockfd, void *buf, int len, unsigned int flags, struct sockaddr *from, int *fromlen);
  - `fromlen` khởi tạo bằng sizeof (struct sockaddr).
  - Dùng cho lập trình socket với UDP

Các hàm dùng cho lập trình socket (5)

- Một số hàm dùng cho việc chuyển đổi
  - u_short **htons(u_short host_short)** // host to network short
  - u_short **ntohs(u_short network_short)** // network to host short
  - u_long **htonl(u_long host_long)** // host to network long
  - u_long **ntohl(u_long network_long)** // network to host long
  - char ***inet_ntoa(struct in_addr inaddr)**
  - int **inet_aton(const char *strptr, struct in_addr *addptr)**
- Một số hàm dùng cho việc thao tác dữ liệu
  - void ***memset (void *dest, int char, int len)**
  - void ***memcopy (void *dest, void *src, int len)**
  - int **memcmp (const void *first, const void *second, int len)**

Lập trình TCP Socket

- Kết nối và gửi dữ liệu theo giao thức TCP được dùng rất nhiều trong các ứng dụng do tính **an toàn** và **tin cậy** của nó
- Ta thử xây dựng một mô hình ứng dụng khách/chủ (client/server) sử dụng lớp Socket và ServerSocket để kết nối và trao đổi thông tin giữ máy khách và máy chủ.

Lập trình TCP Socket – Ví dụ

- Máy khách (client) sẽ gửi một chuỗi ký tự đến máy chủ (server), máy chủ sẽ đảo ngược thứ tự của chuỗi và gửi trả về cho máy khách.
- Để mô hình này hoạt động ta phải xây dựng hai chương trình:
  - một chương trình là EchoServer chạy trên máy chủ dùng để lắng nghe kết nối từ phía máy khách và xử lý yêu cầu do máy khách đưa đến,
  - một chương trình khác là EchoClient chạy trên máy khách có nhiệm vụ nhận dữ liệu nhập vào từ bàn phím sau đó gửi đến cho máy chủ xử lý.
- Khi máy chủ xử lý xong, chương trình EchoClient sẽ nhận về và đưa kết quả ra màn hình.

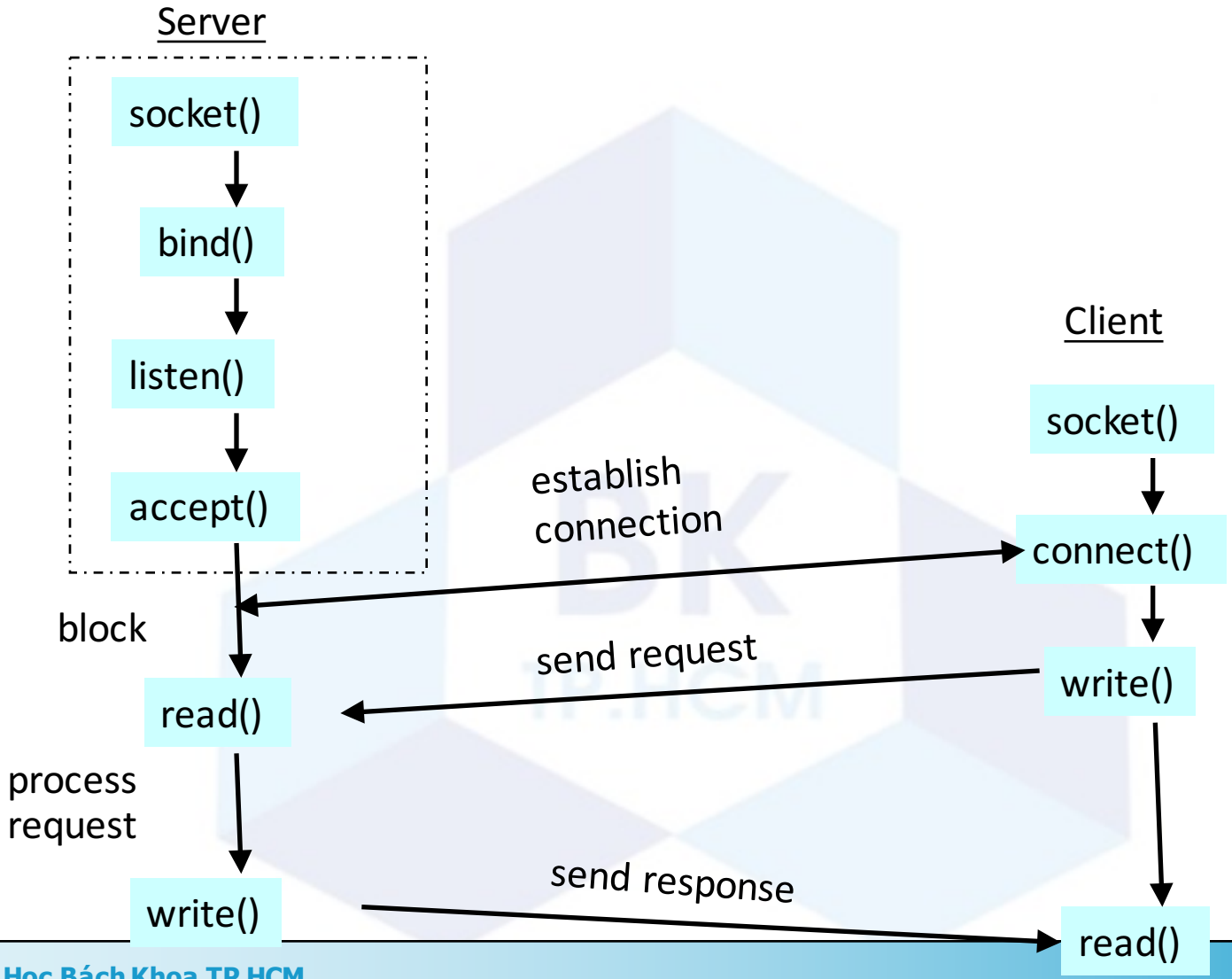
Lập trình UDP Socket

- Mặc dù truyền dữ liệu theo giao thức UDP **không chính xác** và **không đảm bảo bằng TCP** nhưng nó không đòi hỏi nhiều tài nguyên của hệ thống, quá trình xử lý và tiếp nhận dữ liệu cũng đơn giản hơn nhiều.
- Dữ liệu gửi đi theo giao thức UDP thường được đóng thành một gói (data package) bao gồm địa chỉ IP của máy nhận, số cổng cùng với dữ liệu thật sự bên trong.
- UDP được dùng trong các ứng dụng mang tính chất **thông báo** như về giá cả, thời tiết …
- Ta thử xây dựng một mô hình ứng dụng khách/chủ (client/server) sử dụng lớp DatagramSocket và DatagramPackage để kết nối và trao đổi thông tin giữa máy khách và máy chủ bằng giao thức UDP.

Lập trình UDP Socket – Ví dụ

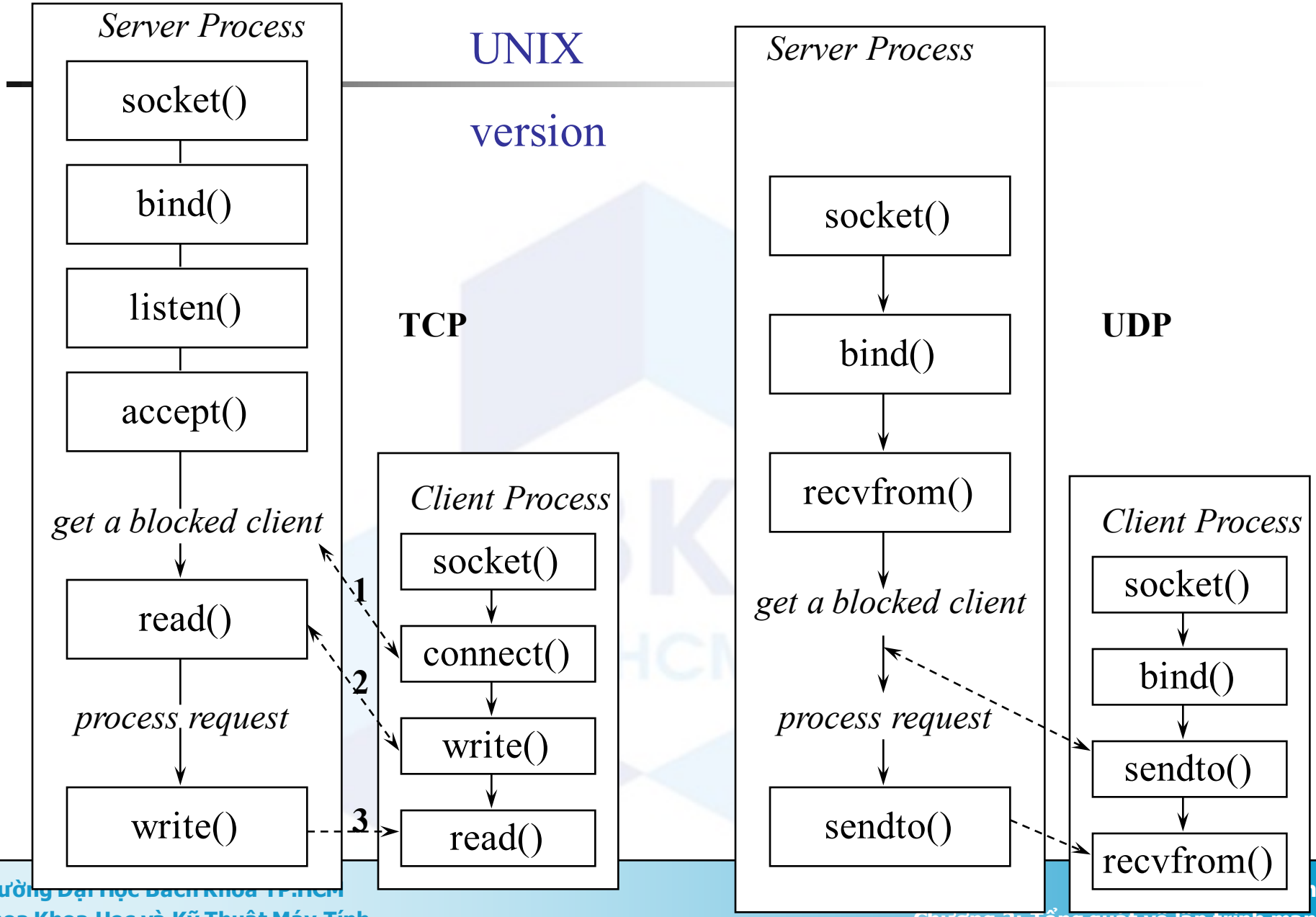
- ExchangeRateServer chạy trên máy chủ và đón nhận những dữ liệu do máy khách gửi đến cổng 2345.
- Khi nhận được yêu cầu, máy chủ sẽ gửi trả các thông báo về tỉ giá, kèm theo ngày giờ mới nhất về cho máy khách.
- Ta sử dụng hàm random của lớp Math để lấy về tỷ giá mang tính chất ngẫu nhiên của 3 thị trường là Tokyo, New York và Hong Kong.
- Dữ liệu được nhận và gửi theo từng gói dựa vào lớp DatagramPackage
- Chương trình chạy trên máy chủ cung cấp tỷ giá của các thị trường chứng khoán.

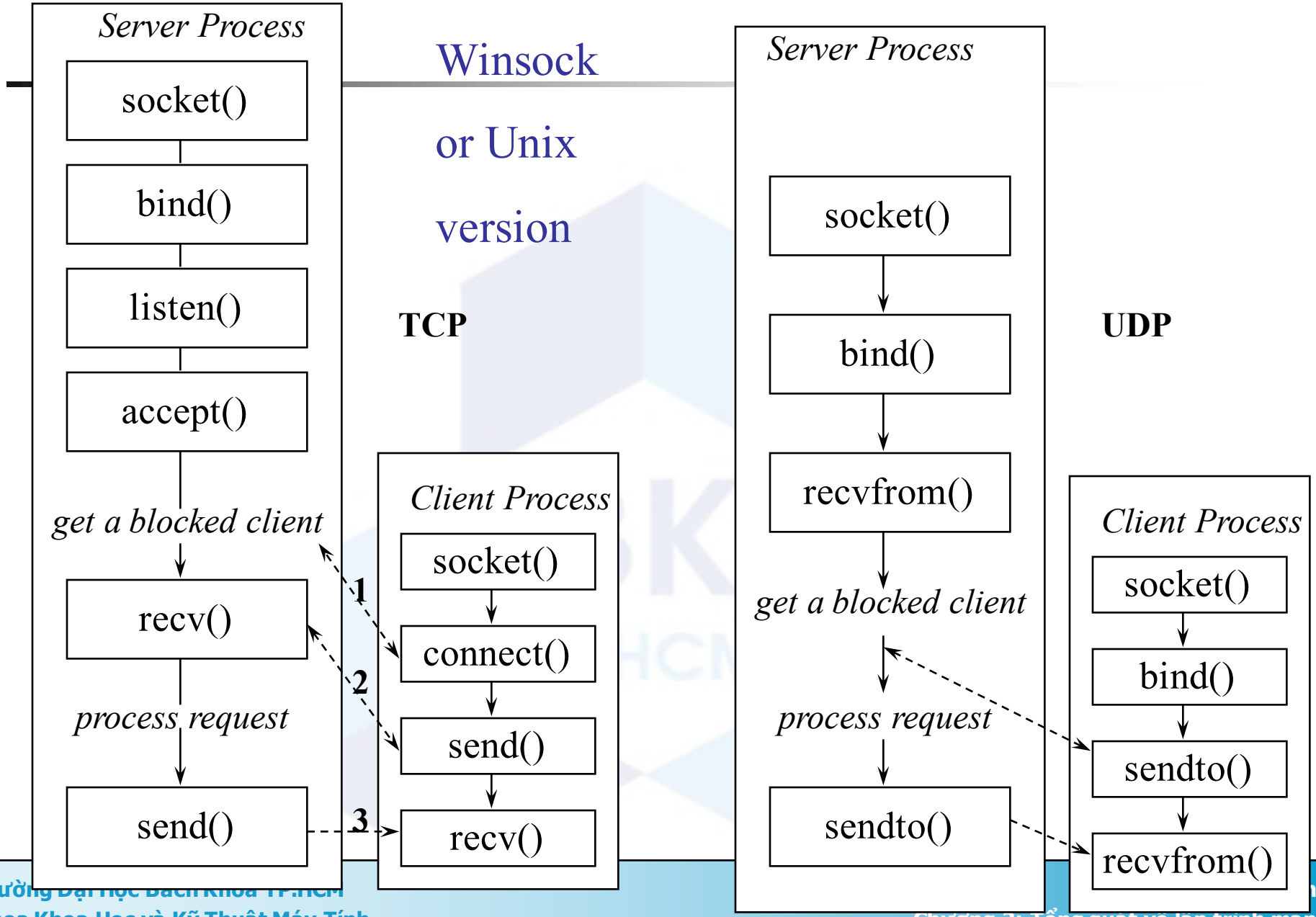
Tổng quan về giao tiếp Socket



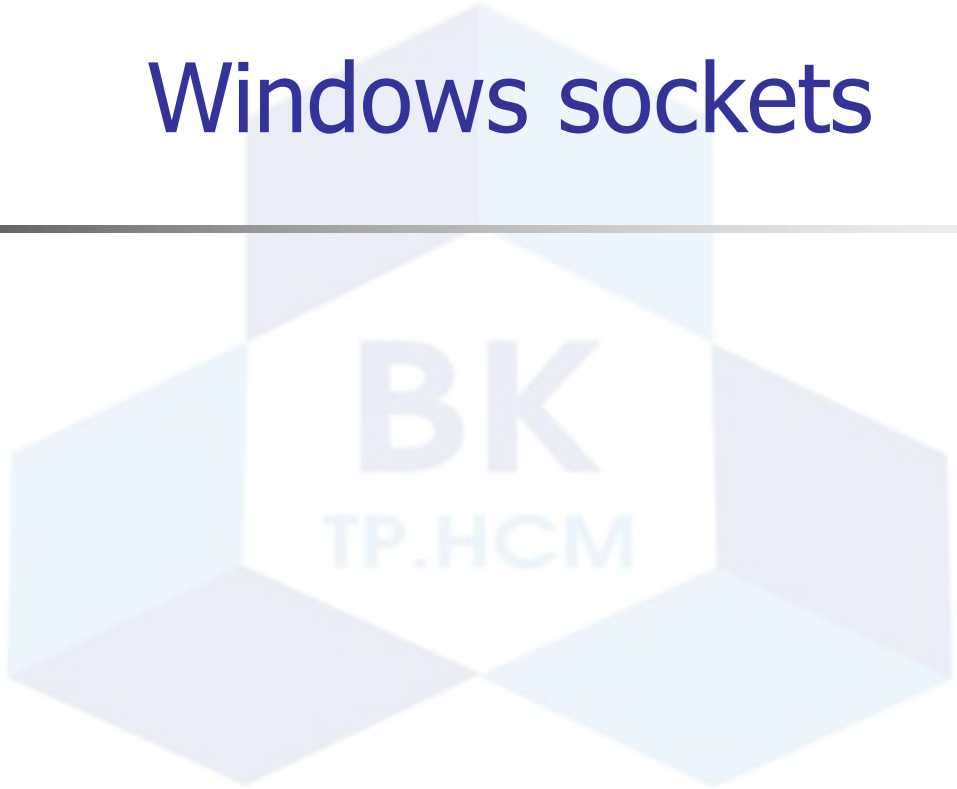
Client: xác định address/port của server

- Server thường được biết đến qua tên (name) và dịch vụ (service)
  - E.g., "**www.cse.hcmut.edu.vn**" và "**http**"
- Cần diễn dịch chúng sang địa chỉ IP và chỉ số port
  - E.g., "**64.236.16.20**" and "**80**"
- Diễn dịch tên của server ra thành địa chỉ
  - `struct hostent *gethostbyname(char *name)`
  - Argument: host name (e.g., "`www.cse.hcmut.edu.vn`")
  - Trả về một giá trị cấu trúc có chứa địa chỉ host
- Xác định port của service
  - `struct servent *getservbyname(char *name, char *protocol)`
  - Arguments: service (e.g., "ftp") và protocol (e.g., "tcp")
  - Static config in `/etc/services`


UNIX
version
Server Process
socket()
bind()
listen()
accept()
get a blocked client
read()
process request
write()
TCP
Client Process
socket()
connect()
write()
read()
1
2
3
Server Process
socket()
bind()
recvfrom()
get a blocked client
process request
sendto()
UDP
Client Process
socket()
bind()
sendto()
recvfrom()


Server Process
socket()
bind()
listen()
accept()
get a blocked client
recv()
process request
send()
Winsock
or Unix
version
TCP
Client Process
socket()
connect()
send()
recv()
1
2
3
Server Process
socket()
bind()
recvfrom()
get a blocked client
process request
sendto()
UDP
Client Process
socket()
bind()
sendto()
recvfrom()

Windows sockets



Các hàm của Winsocks (1)

- Hàm tạo record socket chứa thông tin về cổng giao tiếp :

SOCKET socket (int *af*, int *type*, int *protocol*);

  - af : Họ địa chỉ, thường là AF_INET : Internet
  - type : Kiểu socket (SOCK_STREAM, SOCK_DGRAM)
  - protocol : giao thức được dùng, default = 0
  - return INVALID_SOCKET : error

Ví dụ:

```
// Tạo socket mới, nếu thất bại báo sai
ser_sock=socket(AF_INET,SOCK_STREAM,0);
if(ser_sock==INVALID_SOCKET) {
    MessageBox("Khong tao duoc socket");
    return TRUE;
}
```

Các hàm của Winsocks (2)

- Hàm gắn kết socket với các thông tin trạng thái ban đầu của cổng giao tiếp :

  int **bind** (SOCKET *s*, const struct sockaddr FAR* *name*, int *namelen*);

  - name : record chứa thông tin của cổng giao tiếp
  - namelen : độ dài của record "name"

  return SOCK_ERROR

=SOCKET_ERROR) {
MessageBox("Khong bind socket duoc");
return TRUE;
}

Các hàm của Winsocks (2)

Ví dụ: thiết lập địa chỉ điểm đầu cuối và bind nó với socket

```
SOCKADDR_IN local_addr;
local_addr.sin_family=AF_INET;
local_addr.sin_port=256;
local_addr.sin_addr.s_addr=INADDR_ANY;
if(bind(ser_sock,(LPSOCKADDR)&local_addr,sizeof(local_a
   ddr))==SOCKET_ERROR) {
    MessageBox("Khong bind socket duoc");
    return TRUE;
}
struct sockaddr_in {
    short sin_family; // họ socket Internet
    unsigned short sin_port; // cổng giao tiếp
    struct in_addr sin_addr; // địa chỉ IP của máy
    char sin_zero[8]; // 8 byte 0
};
```

Các hàm của Winsocks (3)

Hàm khai báo độ dài hàng chờ cho các yêu cầu nối kết:

int listen (SOCKET *s*, int *backlog*);

- backlog = nên là SOMAXCONN (Maximum socket connections): độ dài hàng chờ cho các yêu cầu nối kết
- Nếu return < 0: error

Ví dụ:

```
// Khai báo độ dài hàng chờ yêu cầu kết nối
if(listen(ser_sock, SOMAXCONN)==SOCKET_ERROR) {
    MessageBox("Khong listen duoc");
    return TRUE;
}
```

Các hàm của Winsocks (4)

- Hàm lắng nghe và phục vụ yêu cầu kết nối:

SOCKET accept (**SOCKET** *s*, **struct sockaddr FAR*** *addr*, **int FAR*** *addrlen*);

  - addr: record chứa thông tin về cổng từ xa yêu cầu kết nối
  - addrlen: độ dài record "addr"

Ví dụ:

```
SOCKADDR_IN remote_addr;
SOCKET sock;
// Lắng nghe và phục vụ yêu cầu kết nối
int len=sizeof(remote_addr);
sock=accept(ser_sock,(LPSOCKADDR)&remote_addr,&len);
if(sock==INVALID_SOCKET){
    MessageBox("Khong accept duoc");
    return;
}
```

Các hàm của Winsocks (5)

- Hàm khai báo các biến cố bất đồng bộ

int WSAAsyncSelect (SOCKET *s*, **HWND** *hWnd*, **unsigned int** *wMsg*, **long** *lEvent* **);**

  - hWnd : cửa sổ chương trình sẽ nhận message.
  - wMsg : thông báo sẽ tạo ra
  - lEvent : tổ hợp các biến cố network sẽ gây ra thông báo.

Ví dụ:

```
// Khai báo chờ yêu cầu kết nối + nhận data bất đồng bộ
If(WSAAsyncSelect(ser_sock, m_hWnd, WSA_ACCEPT,
   FD_ACCEPT)> 0)
{
    MessageBox("Error on WSAAsyncSelect()");
    closesocket(ser_sock);
}
```

Các hàm của Winsocks (6)

- Hàm yêu cầu kết nối TCP tới server

  int **connect** (**SOCKET** *s*, **const struct sockaddr FAR*** *name*, **int** *namelen*);

  - s: socket địa phương
  - name: record chứa thông tin về cổng giao tiếp từ xa cần nối kết
  - namelen: độ dài của vùng name
  - Nếu return <0: error

Các hàm của Winsocks (7)

- Hàm gởi 1 chuỗi byte đến đối tác

  int **send** (**SOCKET** *s*, **const char FAR** * *buf*, **int***len*, **int** *flags*);

  - buf: pointer tới bộ đệm dữ liệu
  - len: độ dài bộ đệm
  - flags: NO_FLAGS_SET

Các hàm của Winsocks (8)

- Hàm chờ nhận thông tin từ xa gởi tới

  int **recv** (**SOCKET** *s*, **char FAR*** *buf*, **int***len*, **int** *flags*);

  - buf: pointer tới bộ đệm dữ liệu nhận
  - len: độ dài bộ đệm
  - flags: NO_FLAGS_SET
  - return SOCKET_ERROR: error

Các hàm của Winsocks (8)

- Hàm yêu cầu đóng và xóa socket

```
int closesocket(SOCKET s);
```

Tài Liệu Tham Khảo

- [1] *Unix Network Programming, Vol. 1, Third Edition, The Sockets Networking API*, W. Richard Stevens, Bill Fenner, Andrew M. Rudoff, Addison Wesley, 2003
- [2] Bộ Slides cũ môn Lập Trình Mạng, Khoa KH&KTMT.