

Chương 2: Nội dung

2.1. Nguyên lý của ứng dụng mạng

- 2.1.1. Các kiến trúc của ứng dụng mạng
- 2.1.2. Truyền thông giữa các tiến trình
- 2.1.3. Các dịch vụ giao vận

2.2. Web và HTTP

2.3. FTP

2.4. Thư điện tử

2.5. DNS (Domain Name Systems)

2.6. Ứng dụng Peer-to-peer

2.7. Video streaming và các mạng phân phối nội dung

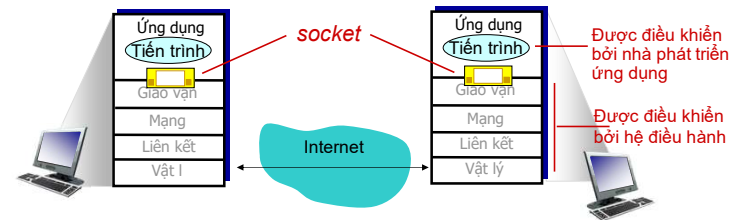
2.8. Lập trình socket với UDP và TCP

Tầng ứng dụng 2-102

Lập trình Socket

Mục đích: hiểu được cách xây dựng ứng dụng truyền thông client/server dùng socket

socket: là cánh cửa giữa các tiến trình ứng dụng và giao thức giao vận end-to-end



Tầng ứng dụng 2-103

Lập trình Socket

Hai loại socket cho hai dịch vụ tầng giao vận:

- **UDP:** truyền các gói tin không tin cậy
- **TCP:** truyền tin cậy, truyền dòng byte có hướng

Ví dụ ứng dụng:

1. Client đọc vào một dòng ký tự (dữ liệu) từ bàn phím và gửi dữ liệu đến server.
2. Server nhận dữ liệu và chuyển các ký tự thành dạng ký tự viết hoa.
3. Server gửi dữ liệu đã được chuyển thành dạng viết hoa về cho client.
4. Client nhận dữ liệu và hiển thị dòng ký tự lên màn hình.

Tầng ứng dụng 2-104

Lập trình socket với UDP

UDP: không có “kết nối” giữa client & server

- ❖ Không bắt tay trước khi gửi dữ liệu
- ❖ Bên gửi gán địa chỉ IP và số hiệu cổng đích vào trong mỗi gói tin
- ❖ Bên nhận sẽ trích địa chỉ IP và số hiệu cổng của bên gửi từ gói tin nhận được

UDP: dữ liệu được truyền có thể bị mất hoặc không đúng trình tự khi nhận

Quan điểm ứng dụng:

- ❖ UDP cung cấp truyền không tin cậy theo các nhóm byte (“các gói tin”) giữa client và server

Tầng ứng dụng 2-105

Tương tác client/server socket: UDP

server (chạy trên serverIP)

Tạo socket, port= x:
`serverSocket =
socket(AF_INET, SOCK_DGRAM)`

Đọc datagram từ
`serverSocket`

Ghi trả lời vào
`serverSocket`
địa chỉ client,
số hiệu cổng
cụ thể

client

Tạo socket:
`clientSocket =
socket(AF_INET, SOCK_DGRAM)`

Tạo datagram với IP của server
và port=x; gửi datagram qua
`clientSocket`

Đọc datagram từ
`clientSocket`
Đóng
`clientSocket`

Tăng ứng dụng 2-106

Ví dụ: UDP client

Python UDPClient

```
include thư viện socket của Python → from socket import *
serverName = 'hostname'
serverPort = 12000

Tạo UDP socket tới server → clientSocket = socket(socket.AF_INET,
                                                    socket.SOCK_DGRAM)

Nhận đầu vào từ bàn phím của người dùng → message = raw_input('Input lowercase sentence:')
Gắn tên, số cổng server vào message; gửi vào socket → clientSocket.sendto(message, (serverName, serverPort))

Đọc các ký tự trả lời từ socket vào chuỗi → modifiedMessage, serverAddress =
                                                    clientSocket.recvfrom(2048)

In ra chuỗi nhận được và đóng socket → print modifiedMessage
                                                    clientSocket.close()
```

Tăng ứng dụng 2-107

Ví dụ: UDP server

Python UDPServer

```
from socket import *
serverPort = 12000
```

Tạo UDP socket → `serverSocket = socket(AF_INET, SOCK_DGRAM)`

Liên kết socket với số cổng cục bộ 12000 → `serverSocket.bind(('', serverPort))`
`print "The server is ready to receive"`

Vòng lặp → while 1:

Đọc từ UDP socket vào message, lấy địa chỉ của client (IP và cổng của client) → `message, clientAddress = serverSocket.recvfrom(2048)`
`modifiedMessage = message.upper()`

Gửi chuỗi ký tự viết hoa cho client này → `serverSocket.sendto(modifiedMessage, clientAddress)`

Tăng ứng dụng 2-108

Lập trình socket với TCP

client phải tiếp xúc với server

- ❖ Tiến trình server phải chạy trước
- ❖ server phải tạo socket (cửa) để đón client tiếp xúc

client tiếp xúc với server bằng cách:

- ❖ Tạo TCP socket, xác định địa chỉ IP, số hiệu cổng của tiến trình server
- ❖ **Khi client tạo socket:** TCP client sẽ thiết lập kết nối tới TCP server

- ❖ Khi được tiếp xúc bởi client, **TCP server sẽ tạo socket mới** cho tiến trình server để truyền thông với client
 - Cho phép server "nói chuyện" với nhiều client
 - Các số hiệu cổng nguồn được dùng để phân biệt các client.

Quan điểm ứng dụng:

TCP cung cấp truyền tin cậy, truyền dòng byte theo đúng trình tự giữa client và server.

Tăng ứng dụng 2-109

Tương tác client/server socket: TCP

server (chạy trên `hostid`)

Tạo socket,
port=`x`, cho yêu cầu
đến:
`serverSocket = socket()`

Đợi yêu cầu kết
nối đến
`connectionSocket =
serverSocket.accept()`

Đọc yêu cầu từ
`connectionSocket`

Ghi trả lời vào
`connectionSocket`

Đóng
`connectionSocket`

client

Tạo socket,
kết nối tới `hostid`, port=`x`
`clientSocket = socket()`

Gửi yêu cầu dùng
`clientSocket`

Đọc trả lời từ
`clientSocket`

Đóng
`clientSocket`

TCP
Thiết lập kết nối

Tăng ứng dụng 2-110

Ví dụ: TCP client

Python TCPClient

```
from socket import *  
serverName = 'servername'  
serverPort = 12000  
clientSocket = socket(AF_INET, SOCK_STREAM)  
clientSocket.connect((serverName, serverPort))  
sentence = raw_input('Input lowercase sentence:')  
clientSocket.send(sentence)  
modifiedSentence = clientSocket.recv(1024)  
print 'From Server:', modifiedSentence  
clientSocket.close()
```

Tạo TCP socket tới
server, remote port 12000

Không cần gán tên, số
cổng server

Tăng ứng dụng 2-111

Ví dụ: TCP server

Python TCPServer

```
from socket import *  
serverPort = 12000  
serverSocket = socket(AF_INET, SOCK_STREAM)  
serverSocket.bind(('', serverPort))  
serverSocket.listen(1)  
print 'The server is ready to receive'  
while 1:  
    connectionSocket, addr = serverSocket.accept()  
    sentence = connectionSocket.recv(1024)  
    capitalizedSentence = sentence.upper()  
    connectionSocket.send(capitalizedSentence)  
    connectionSocket.close()
```

Tạo socket TCP
chờ

server bắt đầu lắng nghe
các yêu cầu TCP đi đến

Vòng lặp

server chờ `accept()` cho các
yêu cầu đến, socket mới
được tạo khi trả về

Đọc các byte từ socket (nhưng
không có địa chỉ như trong UDP)

Đóng kết nối tới client
này

Tăng ứng dụng 2-112

Xây dựng một Web server đơn giản

- ❖ Quản lý một yêu cầu HTTP
- ❖ Chấp nhận yêu cầu
- ❖ Phân tích cú pháp phần tiêu đề (header)
- ❖ Lấy file được yêu cầu từ hệ thống file của server
- ❖ Tạo thông điệp đáp ứng HTTP:
 - các dòng header + file
- ❖ Gửi đáp ứng đến client
- ❖ Sau khi tạo server, có thể yêu cầu file dùng trình duyệt (ví dụ: IE)
- ❖ Xem giáo trình để biết thêm chi tiết

Tăng ứng dụng 2-113

Chương 2: Tổng kết

Trình bày các vấn đề liên quan đến ứng dụng mạng!

- ❖ Kiến trúc của ứng dụng
 - client-server
 - P2P
- ❖ Các yêu cầu dịch vụ của ứng dụng:
 - Truyền tin cậy, băng thông, trễ
- ❖ Mô hình dịch vụ giao vận của Internet
 - Hướng kết nối, truyền tin cậy: TCP
 - Truyền không tin cậy, truyền gói tin: UDP
- ❖ Các giao thức cụ thể:
 - HTTP
 - FTP
 - SMTP, POP, IMAP
 - DNS
 - P2P: BitTorrent
- ❖ Video streaming, CDNs
- ❖ Lập trình socket: TCP socket, UDP socket

Tăng ứng dụng 2-114

Chương 2: Tổng kết

Quan trọng hơn: được học về các giao thức!

- ❖ Trao đổi giữa các thông điệp yêu cầu/đáp ứng:
 - client yêu cầu thông tin hoặc dịch vụ
 - server đáp ứng với dữ liệu, hoặc mã trạng thái
 - ❖ Định dạng thông điệp:
 - Phần tiêu đề (header): các trường với thông tin về dữ liệu
 - Dữ liệu: thông tin được truyền thông
- Các vấn đề quan trọng:**
- ❖ Tập trung và không tập trung
 - ❖ Không trạng thái và có trạng thái
 - ❖ Khả năng mở rộng
 - ❖ Truyền thông điệp tin cậy và truyền thông điệp không tin cậy
 - ❖ “Sự phức tạp tại phần cạnh của mạng”

Tăng ứng dụng 2-115

Tham khảo

- Jim Kurose, Keith Ross, “Computer Networking: A Top-Down Approach” 8th edition, Pearson, 2020.

Tăng ứng dụng 1-116