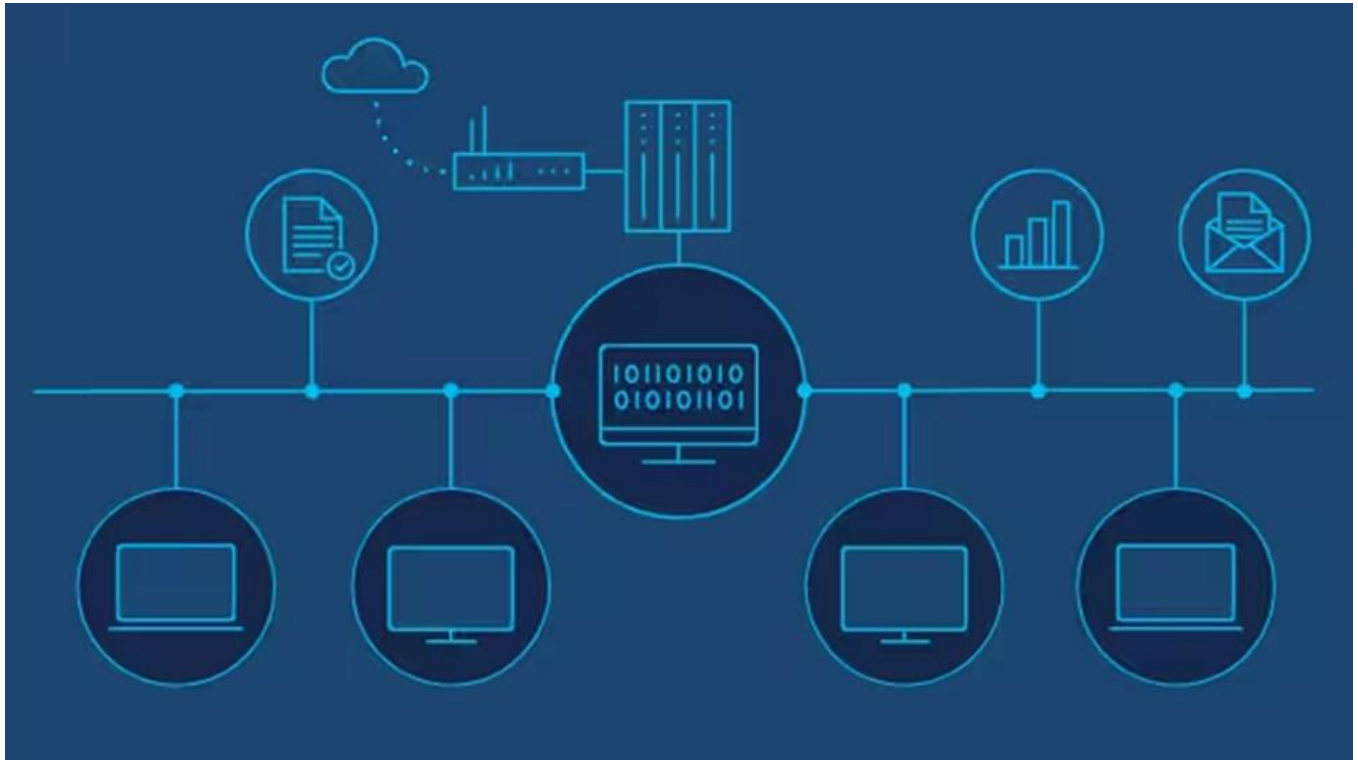
Bài 3
Lập trình Socket với UDP và TCP



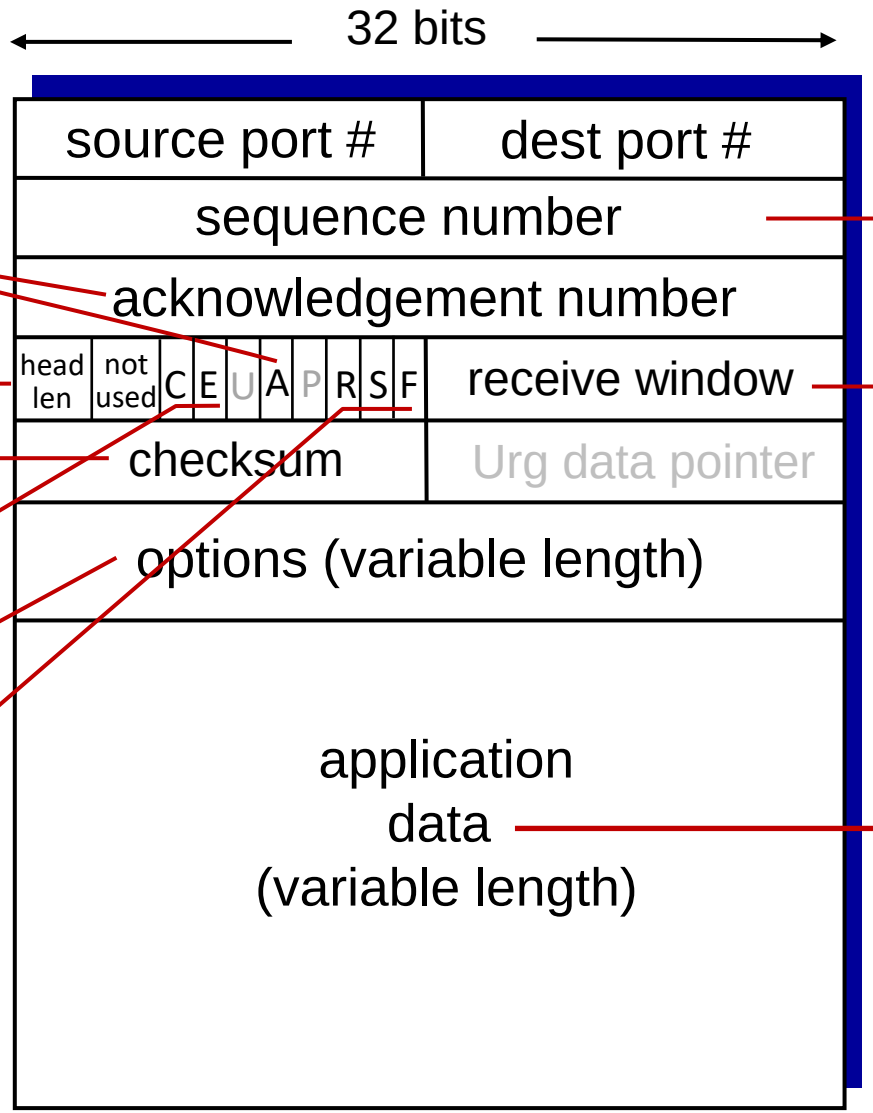
Bài 3: Lộ trình

- Bắt tay ba bước để thiết lập kết nối TCP
- Sử dụng Wireshark trong lập trình mạng
- Hoạt động nhóm

Bài 3: Lộ trình

- Bắt tay ba bước để thiết lập kết nối TCP
- Sử dụng Wireshark trong lập trình mạng
- Hoạt động nhóm

Định dạng phân đoạn TCP



ACK: seq # of next expected byte; A bit: this is an ACK

length (of TCP header)

Internet checksum

C, E: congestion notification

TCP options

RST, SYN, FIN: connection management

segment seq #: counting bytes of data into bytestream (not segments!)

flow control: # bytes receiver willing to accept

Dữ liệu được gửi bởi ứng dụng tới TCP socket

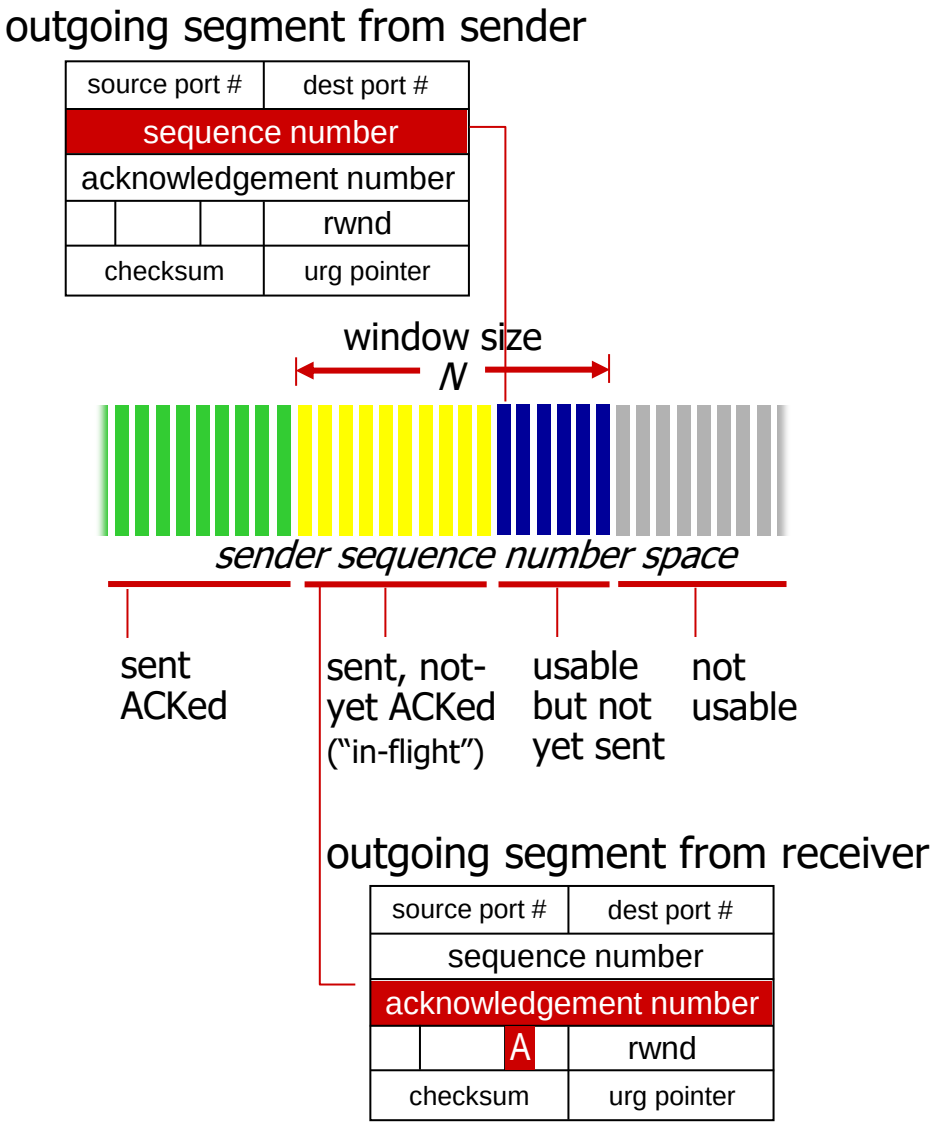
Số tuần tự TCP (TCP sequence numbers), ACKs

Số tuần tự (seq #):

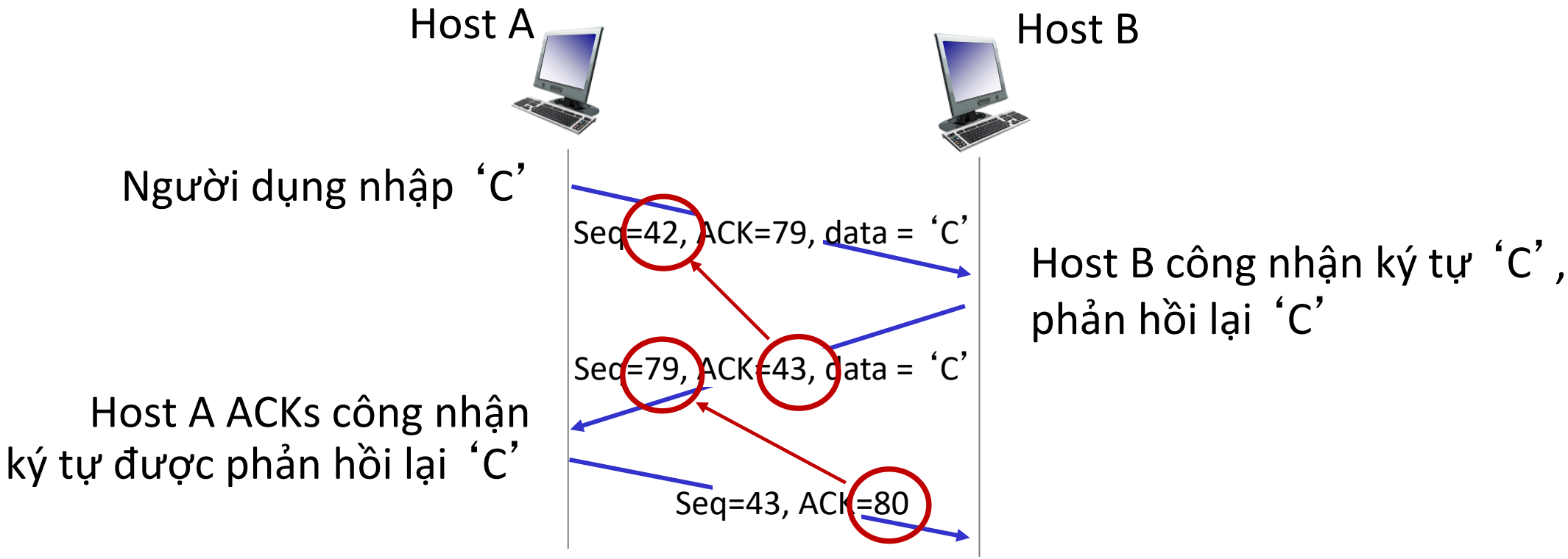
- Trong luồng byte được truyền, số tuần tự là "số" của byte đầu tiên trong dữ liệu của phân đoạn

Công nhận (thông báo nhận):

- Số tuần tự (seq #) của byte kỳ vọng tiếp theo từ bên nhận
- ACK tích lũy



Số tuần tự TCP (TCP sequence numbers), ACKs



Kịch bản telnet đơn giản

Bắt tay 3 bước TCP

Client state

clientSocket = socket(AF_INET, SOCK_STREAM)

LISTEN

clientSocket.connect((serverName,serverPort))

choose init seq num, x
send TCP SYN msg

SYNSENT

SYNbit=1, Seq=x

Server state

serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((‘’,serverPort))
serverSocket.listen(1)
connectionSocket, addr = serverSocket.accept()

LISTEN

choose init seq num, y
send TCP SYNACK
msg, acking SYN

SYN RCVD

SYNbit=1, Seq=y
ACKbit=1; ACKnum=x+1

received SYNACK(x)
ESTAB indicates server is live;
send ACK for SYNACK;
this segment may contain
client-to-server data

ACKbit=1, ACKnum=y+1

received ACK(y)
indicates client is live

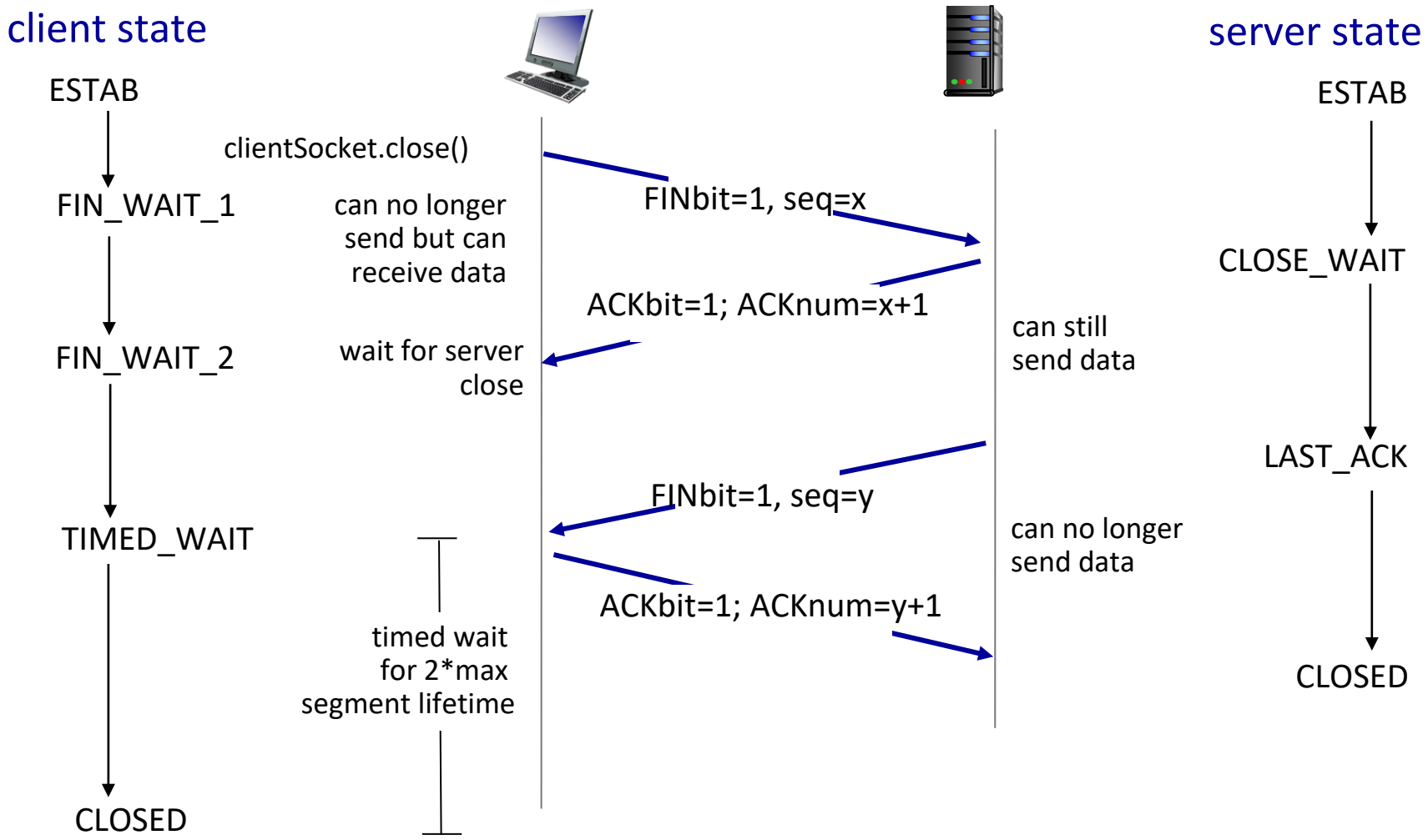
ESTAB

Giao thức bắt tay 3 bước của con người

Đóng kết nối TCP

- Mỗi client, server đóng kết nối phía của chúng
  - Gửi phân đoạn TCP với bit FIN = 1
- Phản hồi FIN được nhận với ACK
  - Khi nhận được FIN, ACK có thể được kết hợp với FIN riêng
- Trao đổi FIN đồng thời có thể được xử lý

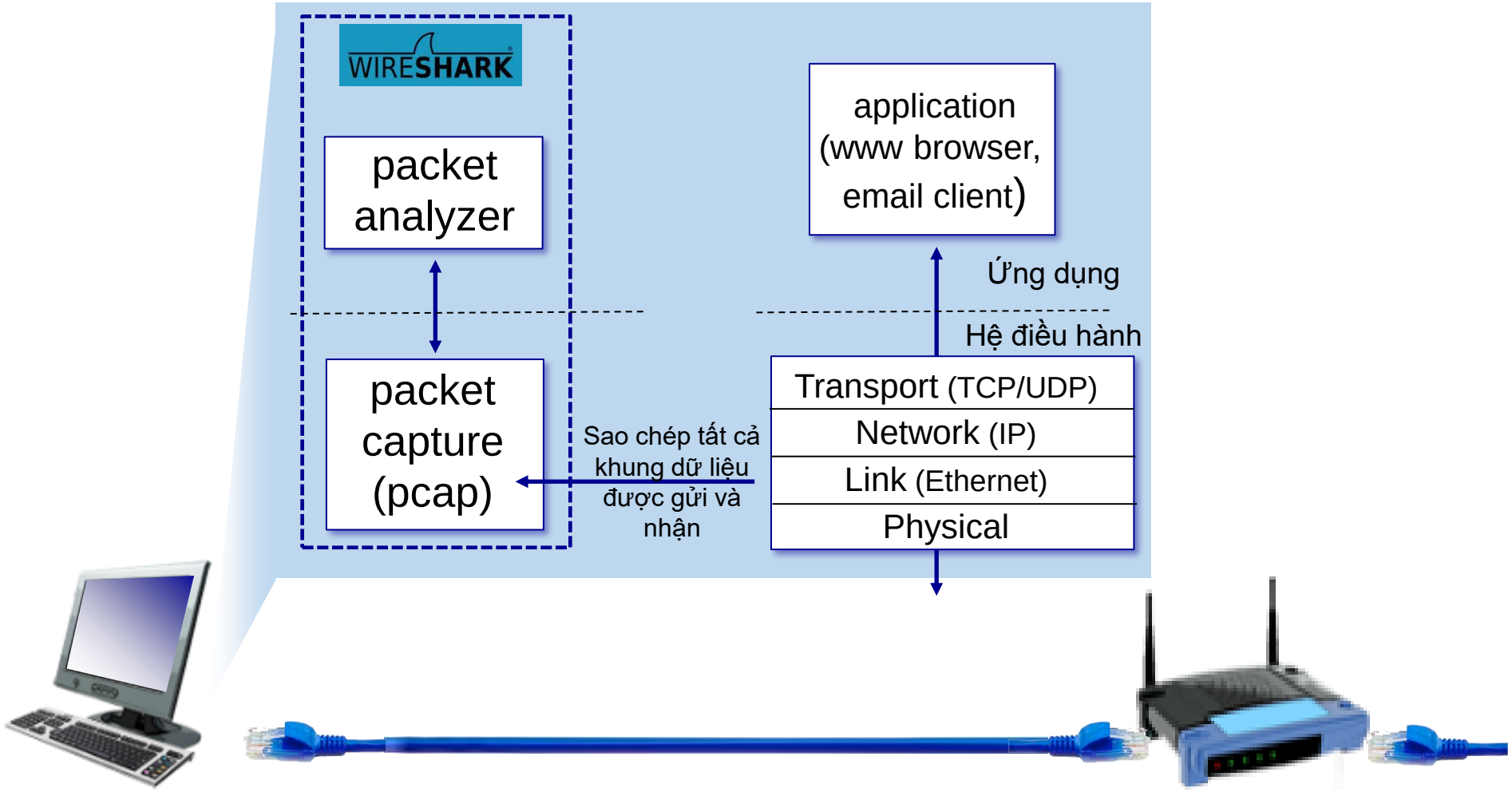
Closing a TCP connection



Bài 3: Lộ trình

- Bắt tay ba bước để thiết lập kết nối TCP
- Sử dụng Wireshark trong lập trình mạng
- Hoạt động nhóm

Sử dụng Wireshark trong lập trình mạng



Sử dụng Wireshark trong lập trình mạng

- Sử dụng WireShark để bắt phân tích gói tin được trao đổi giữa Client và Server đối với 2 hoạt động nhóm trong bài học trước.
- Gợi ý: Sử dụng giao diện "Adapter for loopback traffic capture" để bắt gói tin khi chạy trên localhost (sử dụng địa chỉ IP loopback).

Bài 3: Lộ trình

- Bắt tay ba bước để thiết lập kết nối TCP
- Sử dụng Wireshark trong lập trình mạng
- Hoạt động nhóm

Hoạt động nhóm số 1

- Viết chương trình Server và Client chạy trên localhost để truyền dữ liệu cho nhau. Dữ liệu là văn bản chứa trong tập tin content.txt với nội dung: “Lập trình mạng”. Client gửi văn bản đến Server và yêu cầu Server viết hoa văn bản đó và gửi lại cho Client.
  - Nhóm 1: sử dụng **UDP**
  - Nhóm 2: sử dụng **TCP**

Lời giải: Hoạt động nhóm số 1 - UDP

Server

```
import socket
serverIP = "127.0.0.10"
serverPort = 10000
maxBytes = 4096

sock = socket.socket(socket.AF_INET,
                     socket.SOCK_DGRAM)
sock.bind((serverIP, serverPort))
while True:
  message, clientAddress =
                   sock.recvfrom(maxBytes)
  print("Client address:", clientAddress)

  message = message.decode()
  modifiedMessage = message.upper()
  sock.sendto(modifiedMessage.encode(),
clientAddress)
  pass
```

Client

```
import socket
serverIP = "127.0.0.10"
serverPort = 10000
maxBytes = 4096

with open('content.txt', mode='rt',
encoding='utf-8') as f:
  data = f.read()

sock = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
sock.sendto(data.encode(), (serverIP,
serverPort))
modifiedMessage, serverAddress =
sock.recvfrom(maxBytes)
sock.close()

modifiedMessage = modifiedMessage.decode()
print(modifiedMessage)
```

Lời giải: Hoạt động nhóm số 1 - TCP

Server

```
import socket
serverIP = "127.0.0.10"
serverPort = 10000
maxBytes = 4096

sock = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
sock.bind((serverIP, serverPort))
sock.listen()
while True:
  connectionSocket, address=sock.accept()
  message=connectionSocket.recv(maxBytes)
  print("TCP connection address:",address)

  message = message.decode()
  modifiedMessage = message.upper()
  connectionSocket.send(modifiedMessage.encode())
  pass
```

Client

```
import socket
serverIP = "127.0.0.10"
serverPort = 10000
maxBytes = 4096

with open('content.txt', mode='rt',
encoding='utf-8') as f:
  data = f.read()

sock = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
sock.connect((serverIP, serverPort))

sock.send(data.encode())
modifiedMessage = sock.recv(maxBytes)
sock.close()

modifiedMessage = modifiedMessage.decode()
print(modifiedMessage)
```

Minh hoạ: Hoạt động nhóm số 1 - UDP

*Adapter for loopback traffic capture

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr==127.0.0.10

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.10	UDP	51	61335 → 10000 Len=19
2	0.000235	127.0.0.10	127.0.0.1	UDP	51	10000 → 61335 Len=19

```
> Frame 1: 51 bytes on wire (408 bits), 51 bytes captured (408 bits) on interface \Device\NPF_Loo
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.10
> User Datagram Protocol, Src Port: 61335, Dst Port: 10000
> Data (19 bytes)
```

```
0000  02 00 00 00 45 00 00 2f  30 72 00 00 80 11 00 00   ····E··/ 0r····· 
0010  7f 00 00 01 7f 00 00 0a  ef 97 27 10 00 1b bd 7d   ········ ··'····}
0020  4e 65 74 77 6f 72 6b 20  70 72 6f 67 72 61 6d 6d   Network  programm
0030  69 6e 67                                          ing
```

Minh hoạ: Hoạt động nhóm số 1 - TCP

*Adapter for loopback traffic capture

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr==127.0.0.10

No.	Time	Source	Destination	Protocol	Length	Info
15	3.109046	127.0.0.1	127.0.0.10	TCP	44	62759 → 10000 [ACK] Seq=1 Ack=1 Win=26
16	3.109089	127.0.0.1	127.0.0.10	TCP	63	62759 → 10000 [PSH, ACK] Seq=1 Ack=1 W
17	3.109126	127.0.0.10	127.0.0.1	TCP	44	10000 → 62759 [ACK] Seq=1 Ack=20 Win=2
18	3.109372	127.0.0.10	127.0.0.1	TCP	63	10000 → 62759 [PSH, ACK] Seq=1 Ack=20
19	3.109411	127.0.0.1	127.0.0.10	TCP	44	62759 → 10000 [ACK] Seq=20 Ack=20 Win=
20	3.109503	127.0.0.1	127.0.0.10	TCP	44	62759 → 10000 [FIN, ACK] Seq=20 Ack=20
21	3.109537	127.0.0.10	127.0.0.1	TCP	44	[TCP Dup ACK 17#1] 10000 → 62759 [ACK]
22	3.109562	127.0.0.10	127.0.0.1	TCP	44	10000 → 62759 [ACK] Seq=20 Ack=21 Win=

```
> Frame 18: 63 bytes on wire (504 bits), 63 bytes captured (504 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.10, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 10000, Dst Port: 62759, Seq: 1, Ack: 20, Len: 19
> Data (19 bytes)
```

```
0000  02 00 00 00 45 00 00 3b  9b 49 40 00 80 06 00 00   ····E··; ·I@·····
0010  7f 00 00 0a 7f 00 00 01  27 10 f5 27 87 ae 6b 75   ········ '··'··ku
0020  31 0d 93 cd 50 18 27 f9  a8 f8 00 00 4e 45 54 57   1···P·'· ····NETW
0030  4f 52 4b 20 50 52 4f 47  52 41 4d 4d 49 4e 47      ORK PROG RAMMING
```

Hoạt động nhóm 2

- Write a Server and Client program running on localhost to transfer data to each other. The data is image with the name image.png. Hint: images can be created with Paint software. Image size is less than 4096 bytes. Client sends the image to Server, and then Server save the image in its local storage.
  - Group 1: using **TCP**
  - Group 2: using **UDP**

Lời giải: Hoạt động nhóm số 2 - UDP

Server

```
import socket
serverIP = "127.0.0.10"
serverPort = 10000
maxBytes = 4096

sock = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
sock.bind((serverIP, serverPort))
while True:
  message, clientAddress =
sock.recvfrom(maxBytes)
  print("Client address:", clientAddress)

  with open('image2.png',mode='wb') as f:
    data = f.write(message)
  pass
```

Client

```
import socket
serverIP = "127.0.0.10"
serverPort = 10000
maxBytes = 4096

with open('image.png', mode='rb') as f:
  data = f.read()
  pass

sock = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)

sock.sendto(data, (serverIP,
serverPort))

sock.close()
```

Lời giải: Hoạt động nhóm số 2 - TCP

Server

```
import socket
serverIP = "127.0.0.10"
serverPort = 10000
maxBytes = 4096

sock = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
sock.bind((serverIP, serverPort))
sock.listen()
while True:
  connectionSocket, address=sock.accept()
  message=connectionSocket.recv(maxBytes)
  print("TCP connection address:",address)

  with open('image2.png', mode='wb') as f:
    data = f.write(message)
  pass
```

Client

```
import socket
serverIP = "127.0.0.10"
serverPort = 10000
maxBytes = 4096

with open('image.png', mode='rb') as f:
  data = f.read()
  pass

sock = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)

sock.connect((serverIP, serverPort))

sock.send(data)

sock.close()
```

Minh hoạ: Hoạt động nhóm số 2 - UDP

*Adapter for loopback traffic capture

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr==127.0.0.10

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.10	UDP	2069	63438 → 10000 Len=2037

```
> Frame 1: 2069 bytes on wire (16552 bits), 2069 bytes captured (16552 bits) on interface \Device\
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.10
> User Datagram Protocol, Src Port: 63438, Dst Port: 10000
> Data (2037 bytes)
```

```
0000  02 00 00 00 45 00 08 11  30 7e 00 00 80 11 00 00   ····E··· 0~····
0010  7f 00 00 01 7f 00 00 0a  f7 ce 27 10 07 fd e7 9a   ········ ··'·····
0020  89 50 4e 47 0d 0a 1a 0a  00 00 00 0d 49 48 44 52   ·PNG···· ····IHDR
0030  00 00 01 01 00 00 00 75  08 06 00 00 00 4e 28 54   ·······u ·····N(T
0040  9e 00 00 00 01 73 52 47  42 00 ae ce 1c e9 00 00   ·····sRG B·······
0050  00 04 67 41 4d 41 00 00  b1 8f 0b fc 61 05 00 00   ··gAMA·· ····a···
0060  00 09 70 48 59 73 00 00  0e c2 00 00 0e c2 01 15   ··pHYs·· ········
```

Minh hoạ: Hoạt động nhóm số 2 - TCP

*Adapter for loopback traffic capture

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr==127.0.0.10

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.10	TCP	56	62755 → 10000 [SYN] Seq=0 Win=65535 Le
2	0.000050	127.0.0.10	127.0.0.1	TCP	56	10000 → 62755 [SYN, ACK] Seq=0 Ack=1 W
3	0.000102	127.0.0.1	127.0.0.10	TCP	44	62755 → 10000 [ACK] Seq=1 Ack=1 Win=26
4	0.000231	127.0.0.1	127.0.0.10	TCP	2081	62755 → 10000 [PSH, ACK] Seq=1 Ack=1 W
5	0.000295	127.0.0.10	127.0.0.1	TCP	44	10000 → 62755 [ACK] Seq=1 Ack=2038 Win
6	0.000323	127.0.0.1	127.0.0.10	TCP	44	62755 → 10000 [FIN, ACK] Seq=2038 Ack=
7	0.000348	127.0.0.10	127.0.0.1	TCP	44	[TCP Dup ACK 5#1] 10000 → 62755 [ACK]
8	0.000398	127.0.0.10	127.0.0.1	TCP	44	10000 → 62755 [ACK] Seq=1 Ack=2039 Win

```
> Frame 4: 2081 bytes on wire (16648 bits), 2081 bytes captured (16648 bits) on interface \Device\NPF_Loopback, i
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.10
> Transmission Control Protocol, Src Port: 62755, Dst Port: 10000, Seq: 1, Ack: 1, Len: 2037
> Data (2037 bytes)
```

```
0020  68 21 7a 0d 50 18 27 f9  ee fe 00 00 89 50 4e 47   h!z·P·'· ·····PNG
0030  0d 0a 1a 0a 00 00 00 0d  49 48 44 52 00 00 01 01   ········ IHDR····
0040  00 00 00 75 08 06 00 00  00 4e 28 54 9e 00 00 00   ···u···· ·N(T····
0050  01 73 52 47 42 00 ae ce  1c e9 00 00 00 04 67 41   ·sRGB··· ·····gA
```

Hoạt động nhóm số 3

- Nhóm 1: Đóng vai trò là điểm phát sóng di động và chạy chương trình Server trong **Hoạt động nhóm số 1** sử dụng **TCP**
- Nhóm 2: Kết nối Wi-Fi tới Nhóm 1 và chạy chương trình Client sử dụng **TCP**
- Nhóm 3: Đóng vai trò là điểm phát sóng di động và chạy chương trình Server trong **Hoạt động nhóm số 2** sử dụng **TCP**
- Nhóm 4: Kết nối Wi-Fi tới Nhóm 3 và chạy chương trình Client sử dụng **TCP**
- Nhóm 5: Đóng vai trò là điểm phát sóng di động và chạy chương trình Server trong **Hoạt động nhóm số 1** sử dụng **UDP**
- Nhóm 6: Kết nối Wi-Fi tới Nhóm 5 và chạy chương trình Client sử dụng **UDP**
- Nhóm 7: Đóng vai trò là điểm phát sóng di động và chạy chương trình Server trong **Hoạt động nhóm số 2** sử dụng **UDP**
- Nhóm 8: Kết nối Wi-Fi tới Nhóm 7 và chạy chương trình Client sử dụng **UDP**

Lời giải: Hoạt động nhóm số 3

- Trong cả hai Hoạt động nhóm số 1 và 2, **serverIP** là địa chỉ IP của máy tính đóng vai trò là điểm phát sóng di động!

Bài 3: Tổng kết

- Bắt tay ba bước để thiết lập kết nối TCP
- Sử dụng Wireshark trong lập trình mạng
- Hoạt động nhóm

Thank you