Experiment No. 3

Title: TCP Header Implementation

Batch: B1 **Roll No.:16010420133** **Experiment No.3**

Aim: Write a program to implement header of Transmission Control Protocol segment.

Resources Used: Java / Turbo C/Python/C++

Theory:

The transport service is implemented by a transport protocol used between the two transport entities. Transport protocols resemble the data link protocols. Both have to deal with error control, sequencing, and flow control, among other issues. Differences are due to major dissimilarities between the environments in which the two protocols operate.
The Internet has two main protocols in the transport layer,
Connectionless protocol: UDP
Connection-oriented protocol: TCP

TCP (Transmission Control Protocol) was designed to provide a reliable end-to-end byte stream over an unreliable internetwork .TCP was designed to dynamically adapt to properties of the internetwork and to be robust in the face of many kinds of failures. The IP layer gives no guarantee that datagrams will be delivered properly, so it is up to TCP to time out and retransmit them as need be. Datagrams that do arrive may do so in the wrong order; it is also up to TCP to reassemble them into messages in the proper sequence. TCP must furnish the reliability that most users want and that IP does not provide.
When an application (e.g., a user) process wishes to set up a connection to a remote application process, it must specify which one to connect to. The method normally used is to define transport addresses to which processes can listen for connection requests. In the Internet, these end points are called **ports**.
TCP service is obtained by both the sender and receiver creating end points, called **sockets.**
Each socket has a socket number (address) consisting of the IP address of the host and a 16-bit number local to that host, called a port. A port is the TCP name for a TSAP.
For TCP service to be obtained, a connection(virtual) must be explicitly established between a socket on the sending machine and a socket on the receiving machine. All the segments belonging to a message are sent over this virtual path. A socket may be used for multiple connections at the same time. two or more connections may terminate at the same socket. Port numbers below 256 are called well-known ports and are reserved for standard services. For example, any process wishing to establish a connection to a host to transfer a file using FTP can connect to the destination host's port 21 to contact its FTP daemon. All TCP connections are full duplex and point-to-point.TCP does not support multicasting or broadcasting.
Some assigned ports are:

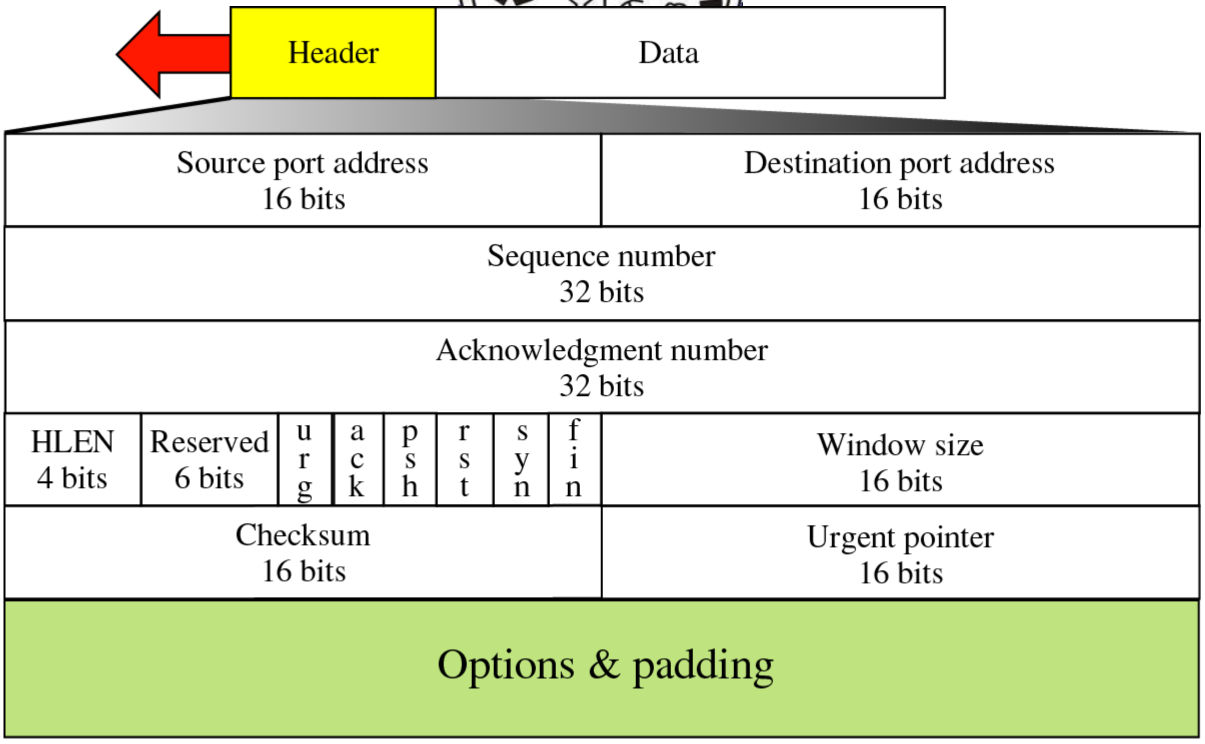
Port	Protocol	Use
21	FTP	File Transfer
23	Telnet	Remote Login
25	SMTP	Email
80	HTTP	World Wide Web

The TCP Protocol:
Every byte on a TCP connection has its own 32-bit sequence number. Separate 32-bit sequence numbers are used for acknowledgements and for the window mechanism. The sending and receiving TCP entities exchange data in the form of segments. A TCP segment consists of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes. Two limits restrict the segment size. Each segment, including the TCP header, must fit in the 65,515-byte IP payload. Each network has a Maximum Transfer Unit, or MTU, and each segment must fit in the MTU. In practice, the MTU is generally 1500 bytes (the Ethernet payload size) and thus defines the upper bound on segment size. A segment that is too large for a n/w can be broken into multiple segments by a router.

The basic protocol used by TCP entities is the **sliding window protocol**. When a sender transmits a segment, it also starts a timer. When the segment arrives at the destination, the receiving TCP entity sends back a segment (with data if any exist, otherwise without data) bearing an acknowledgement number equal to the next sequence number it expects to receive. If the sender's timer goes off before the acknowledgement is received, the sender transmits the segment again.

The TCP Segment Header:

The **Source port** and **Destination port** fields identify the local end points of the connection. A port plus its host's IP address forms a 48-bit unique end point (TSAP). The **Sequence number** defines the number of the first data byte contained in that segment and **Acknowledgement number** specifies the next byte expected, not the last byte correctly received. Both are 32 bits long. The **TCP header length** tells how many 32-bit words are contained in the TCP header.

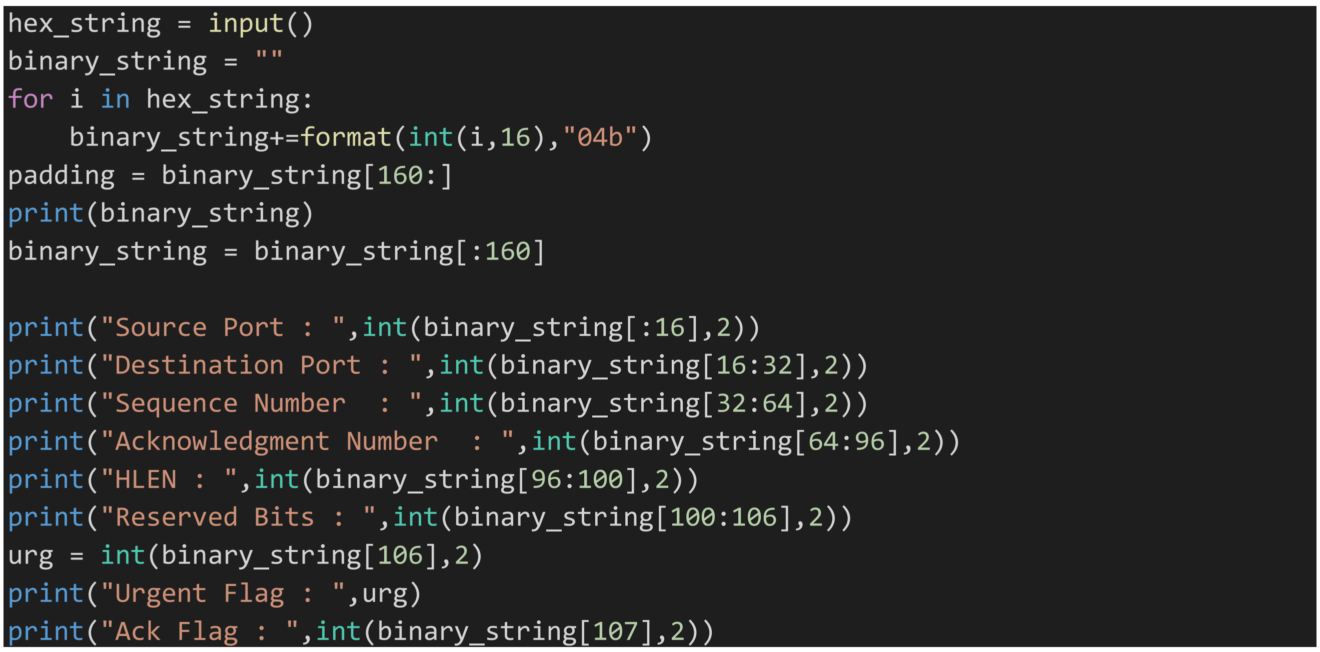


Now come six 1-bit flags.

1. **URG** is set to 1 if the Urgent pointer is in use. The **Urgent pointer** is used to indicate a byte offset from the current sequence number at which urgent data are to be found.
2. The **ACK** bit set to 1 to indicate that the Acknowledgement number is valid. If ACK is 0, the segment does not contain an acknowledgement so the Acknowledgement number field is ignored.
3. The **PSH** bit indicates PUSHed data. Applications can use the PUSH flag, which tells TCP not to delay the transmission.
4. The **RST** bit used to reset a connection that has become confused due to a host crash or some other reason. It is also used to reject an invalid segment or refuse an attempt to open a connection. if you get a segment with the RST bit on, you have a problem on your hands.
5. The **SYN** bit is used to establish connections. The connection request has SYN = 1 and ACK = 0 to indicate that the piggyback acknowledgement field is not in use. The connection reply bears an acknowledgement it has SYN = 1 and ACK = 1. In essence the SYN bit is used to denote CONNECTION REQUEST and CONNECTION ACCEPTED, with the ACK bit used to distinguish between those two possibilities.
6. The **FIN** bit is used to release a connection. It specifies that the sender has no more data to transmit. Both **SYN** and **FIN** segments have sequence numbers and are thus guaranteed to be processed in the correct order.
7. Flow control in TCP is handled using a variable-sized sliding window.
8. The **Window size** field tells how many bytes may be sent starting at the byte acknowledged.

A **Checksum** is also provided for extra reliability. The **Options** field provides a way to add extra facilities not covered by the regular header. There can be upto 40 bytes of optional information in the TCP header The most important option is the one that allows each host to specify the maximum TCP payload it is willing to accept.

Program:



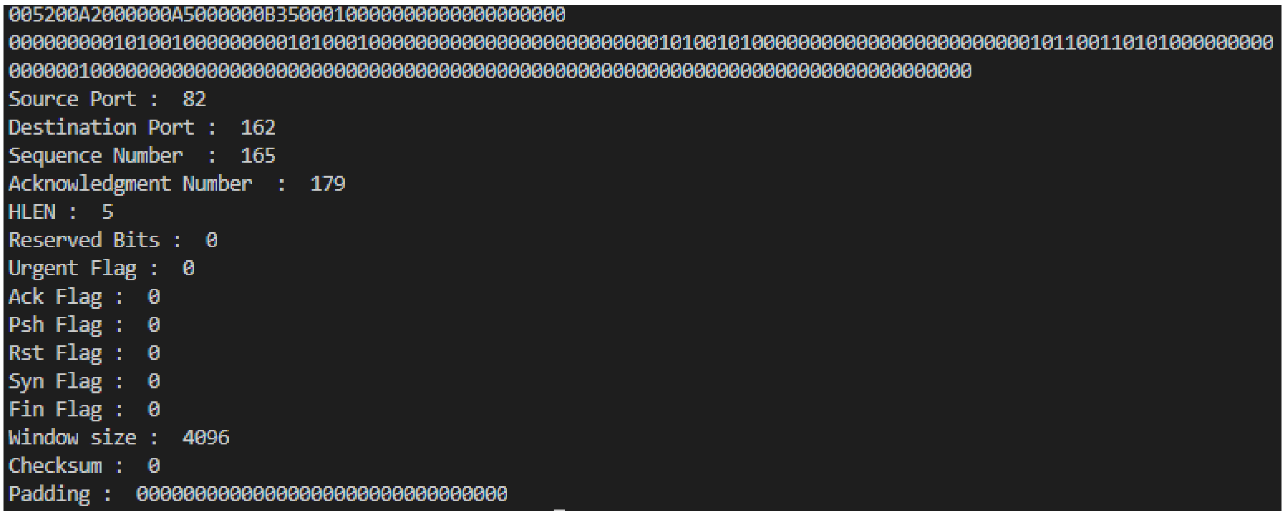
```
hex_string = input()
binary_string = ""
for i in hex_string:
    binary_string+=format(int(i,16),"04b")
padding = binary_string[160:]
print(binary_string)
binary_string = binary_string[:160]

print("Source Port : ",int(binary_string[:16],2))
print("Destination Port : ",int(binary_string[16:32],2))
print("Sequence Number  : ",int(binary_string[32:64],2))
print("Acknowledgment Number  : ",int(binary_string[64:96],2))
print("HLEN : ",int(binary_string[96:100],2))
print("Reserved Bits : ",int(binary_string[100:106],2))
urg = int(binary_string[106],2)
print("Urgent Flag : ",urg)
print("Ack Flag : ",int(binary_string[107],2))
```

```python
print("Psh Flag : ",int(binary_string[108],2))
print("Rst Flag : ",int(binary_string[109],2))
print("Syn Flag : ",int(binary_string[110],2))
print("Fin Flag : ",int(binary_string[111],2))
print("Window size : ",int(binary_string[111:128],2))
print("Checksum : ",int(binary_string[128:144],2))
if(urg):
    print("Urgent Pointer : ",int(binary_string[144:160],2))

print("Padding : ",padding)
```

OUTPUT:



```
005200A2000000A5000000B3500010000000000000000000
00000000010100100000000010100010000000000000000000000000101001010000000000000000000000001011001101010000000000
0000010000000000000000000000000000000000000000000000000000000000000000000000000000
Source Port :  82
Destination Port :  162
Sequence Number  :  165
Acknowledgment Number  :  179
HLEN :  5
Reserved Bits :  0
Urgent Flag :  0
Ack Flag :  0
Psh Flag :  0
Rst Flag :  0
Syn Flag :  0
Fin Flag :  0
Window size :  4096
Checksum :  0
Padding :  00000000000000000000000000000000
```

Questions:

1) In________________services, connection establishment and termination is required.

a) Connection –Oriented
b) Connectionless
c) Segmentation
d) None of the above

Ans: a

2) The unit of data transfer between two devices using TCPis called________________.

Ans: Packet or Frames

3) Which type of addressing is used at Transport Layer?
   a) Port addressing
   b) Logical addressing
   c) Physical Addressing
   d) None of the Above

Ans: a

4) What is flow Control?

In a network, the sender sends the data and the receiver receives the data. But suppose a situation where the sender is sending the data at a speed higher than the receiver is able to receive and process it, then the data will get lost. **Flow-control** methods will help in ensuring this. The flow control method will keep a check that the senders send the data only at a speed that the receiver is able to receive and process.

5) Name the two categories of Congestion Control mechanisms.

Open loop congestion control and Closed loop congestion control.

Outcomes:

Understood the data communication systems, network topologies and network devices.

Conclusion:

With the help of Python, the desired output was successfully achieved.

Grade: AA / AB / BB / BC / CC / CD /DD

References:

Books/ Journals/ Websites:

- B. A. Forouzan and Firouz Mosharraf, "Computer Networks ", A Top-Down Approach ,Special Indian Edition 2012, Tata McGraw Hill.
- Behrouz A Forouzan, TCP/IP Protocol Suite, Tata Mc Graw hill, India, 4th Edition

Signature of faculty in-charge with date

Grade: AA / AB / BB / BC / CC / CD /DD