GITAM
(DEEMED TO BE UNIVERSITY)
VISAKHAPATNAM HYDERABAD BENGALURU

COMPUTER NETWORK LAB RECORD

NAME:	S.Jaswanth				
REDG NO:	122010325012	YEAR:	2021-2022		
BRANCH:	CSE -CS	SEM:	4TH SEM	SEC:	B25
COURSE:	CN LAB	COURSE CODE:	19ECS232P		

EXPERIMENT - 1

<u>Objective:</u> Write a report that includes a diagram showing the topology, type of connection devices, and speed of the wired and wireless LAN in your organization. Also find out the MAC and IP addresses and the subnet mask of your computer.

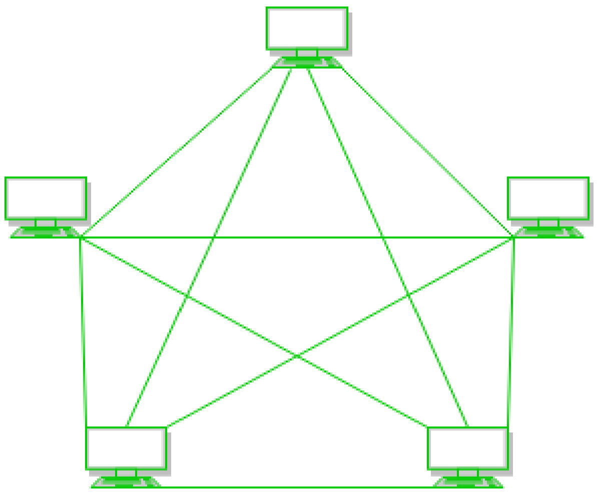
1.Topology:

The arrangement of a network that comprises nodes and connecting lines via sender and receiver is referred to as network topology. Topology simply means that how the end devices are connected to each other.

Types of Topologies:

A) MESH TOPOLOGY:

In a mesh topology, every device is connected to every another device via a particular channel. These channels are known as links.



If N number of devices is connected with each other in a mesh topology, the total number of links that are required by each device is N-1. In the Figure, there are 5 devices connected to each other, hence the total number of links required by each device is 4. Total number of links required=N*(N-1)/2.

- Suppose, N number of devices are connected with each other in a mesh topology, then the total number of dedicated links required to connect them is ${}^{N}C_2$ i.e., N(N-1)/2. In In the

Figure, there are 5 devices connected to each other, hence the total number of links required is 5*4/2 = 10.

Advantages of this topology:

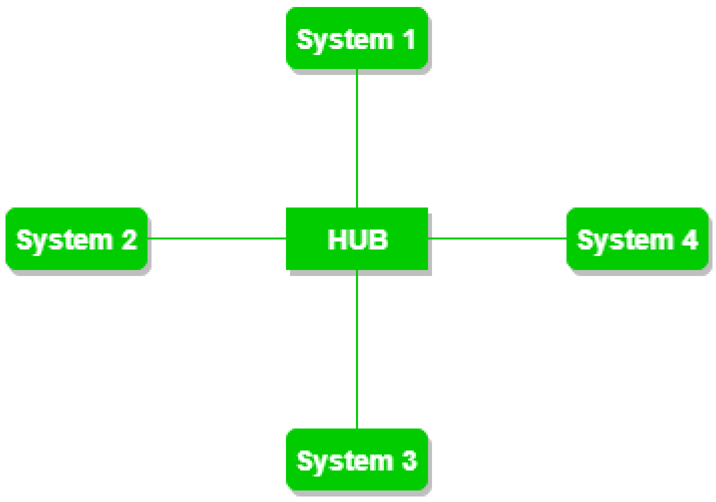
- It is robust.
- The fault is diagnosed easily. Data is reliable because data is transferred among the devices through dedicated channels or links.
- Provides security and privacy.

Problems with this topology:

- Installation and configuration are difficult.
- The cost of cables is high as bulk wiring is required, hence suitable for less number of devices.
- The cost of maintenance is high.

B) STAR TOPOLOGY:

In star topology, all the devices are connected to a single hub through a cable. This hub is the central node and all other nodes are connected to the central node. The hub can be passive in nature i.e., not an intelligent hub such as broadcasting devices, at the same time the hub can be intelligent known as an active hub. Active hubs have repeaters in them.



Advantages of this topology:

- If N devices are connected to each other in a star topology, then the number of cables required to connect them is N. So, it is easy to set up.
- Each device requires only 1 port i.e. to connect to the hub, therefore the total number of ports required is N.

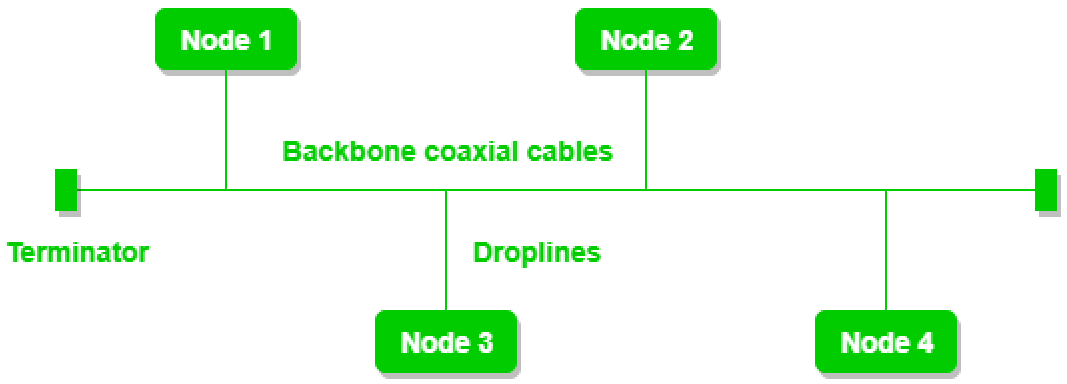
Problems with this topology:

- If the concentrator (hub) on which the whole topology relies fails, the whole system will crash down.

- The cost of installation is high.
- Performance is based on the single concentrator i.e. hub.

C) BUS TOPOLOGY:

Bus topology is a network type in which every computer and network device is connected to a single cable. It transmits the data from one end to another in a single direction. No bi-directional feature is in bus topology. It is a multi-point connection and a non-robust topology because if the backbone fails the topology crashes.



Advantages of this topology:

- If N devices are connected to each other in a bus topology, then the number of cables required to connect them is 1, which is known as backbone cable, and N drop lines are required.
- The cost of the cable is less as compared to other topologies, but it is used to build small networks.

Problems with this topology:

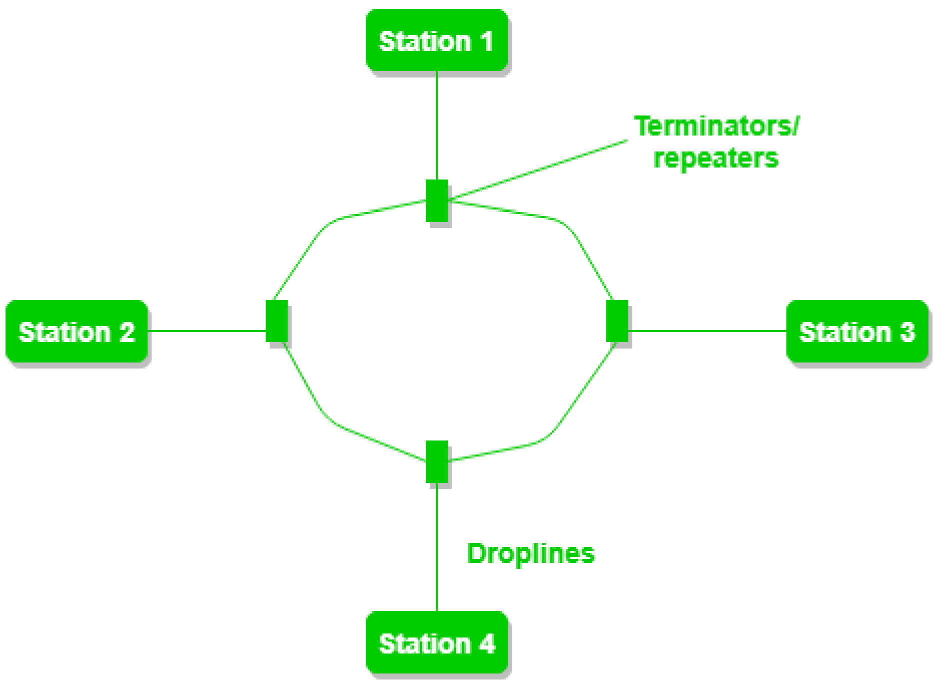
- If the common cable fails, then the whole system will crash down.
- If the network traffic is heavy, it increases collisions in the network. To avoid this, various protocols are used in the MAC layer known as Pure Aloha, Slotted Aloha, CSMA/CD, etc.
- Security is very low.

D) RING TOPOLOGY:

In this topology, the connected devices form a ring with its exactly two neighboring devices.

A number of repeaters are used for Ring topology with a large number of nodes, because if someone wants to send some data to the last node in the ring topology with 100 nodes, then the data will have to pass through 99 nodes to reach the 100th node. Hence to prevent data loss repeaters are used in the network.

The transmission is unidirectional, but it can be made bidirectional by having 2 connections between each Network Node, it is called Dual Ring Topology.



The following operations take place in ring topology are :

1. One station is known as a **monitor** station which takes all the responsibility to perform the operations.
2. To transmit the data, the station has to hold the token. After the transmission is done, the token is to be released for other stations to use.
3. When no station is transmitting the data, then the token will circulate in the ring.
4. There are two types of token release techniques: **Early token release** releases the token just after transmitting the data and **Delay token release** releases the token after the acknowledgment is received from the receiver.

Advantages of this topology:

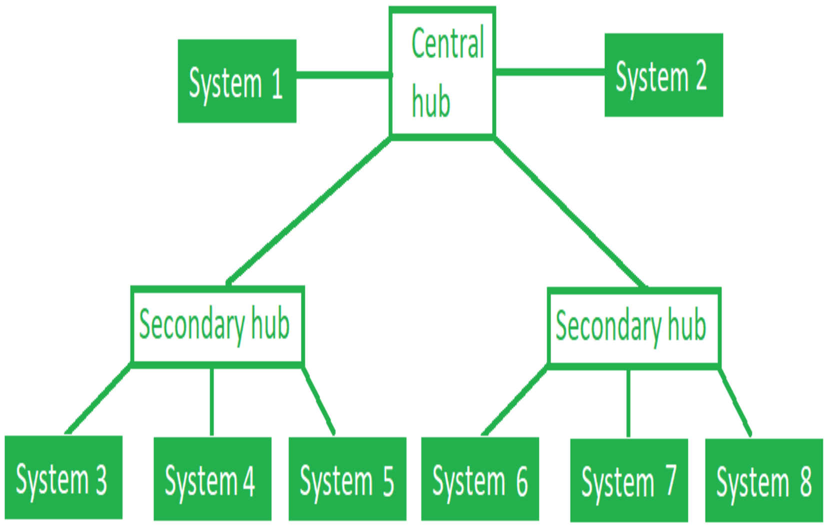
- The possibility of collision is minimum in this type of topology.
- Cheap to install and expand.

Problems with this topology:

- Troubleshooting is difficult in this topology.
- The addition of stations in between or removal of stations can disturb the whole topology.
- Less secure.

E) TREE TOPOLOGY:

This topology is the variation of Star topology. This topology has a hierarchical flow of data.



In this, the various secondary hubs are connected to the central hub which contains the repeater. In this data flow from top to bottom i.e. from the central hub to secondary and then to the devices or from bottom to top i.e. devices to the secondary hub and then to the central hub. It is a multi-point connection and a non-robust topology because if the backbone fails the topology crashes.

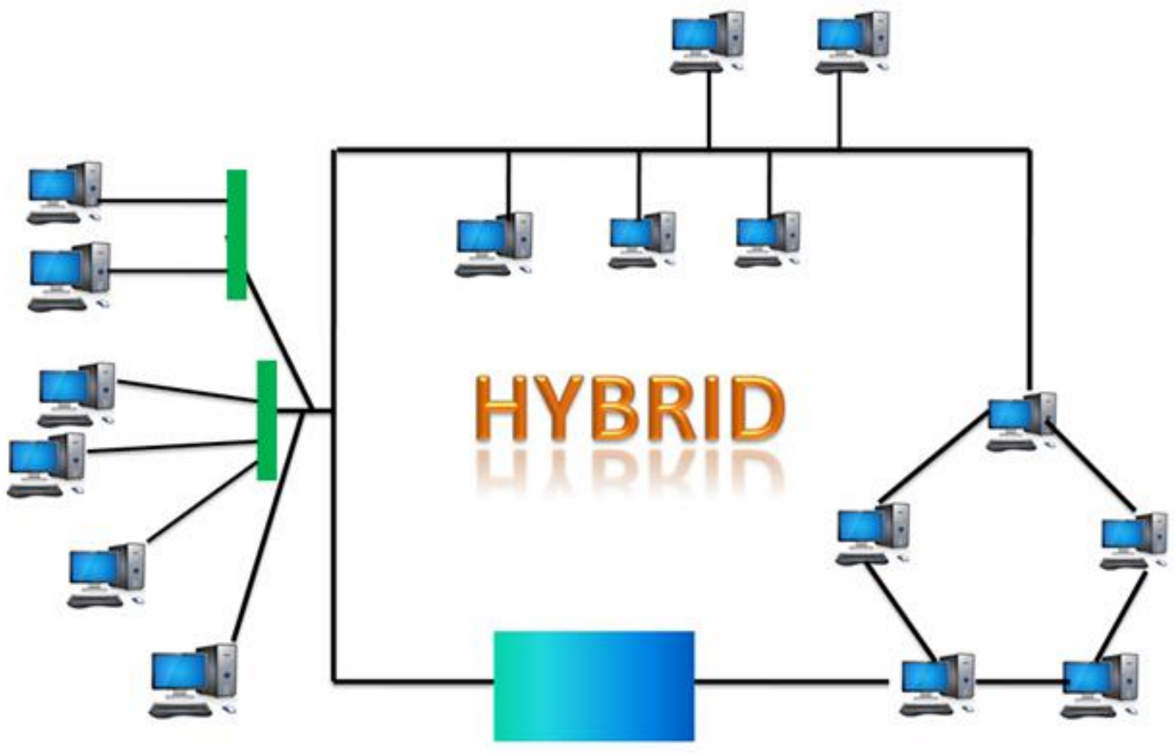
Advantages of this topology:

- It allows more devices to be attached to a single central hub thus it decreases the distance that is traveled by the signal to come to the devices.
- It allows the network to get isolate and also prioritize from different computers.

Problems with this topology:

- If the central hub gets fails the entire system fails.
- The cost is high because of cabling.

F) HYBRID TOPOLOGY



- The combination of various different topologies is known as **Hybrid topology**.
- A Hybrid topology is a connection between different links and nodes to transfer the data.
- When two or more different topologies are combined together is termed as Hybrid topology and if similar topologies are connected with each other will not result in Hybrid topology. For example, if there exists a ring topology in one branch of ICICI bank and bus topology in another branch of ICICI bank, connecting these two topologies will result in Hybrid topology.

Advantages of Hybrid Topology

- **Reliable:** If a fault occurs in any part of the network will not affect the functioning of the rest of the network.
- **Scalable:** Size of the network can be easily expanded by adding new devices without affecting the functionality of the existing network.
- **Flexible:** This topology is very flexible as it can be designed according to the requirements of the organization.
- **Effective:** Hybrid topology is very effective as it can be designed in such a way that the strength of the network is maximized and weakness of the network is minimized.

Disadvantages of Hybrid topology

- **Complex design:** The major drawback of the Hybrid topology is the design of the Hybrid network. It is very difficult to design the architecture of the Hybrid network.
- **Costly Hub:** The Hubs used in the Hybrid topology are very expensive as these hubs are different from usual Hubs used in other topologies.
- **Costly infrastructure:** The infrastructure cost is very high as a hybrid network requires a lot of cabling, network devices, etc.

2.Connection Devices

Physical Layer Devices:

Repeater – A repeater operates at the physical layer. Its job is to regenerate the signal over the same network before the signal becomes too weak or corrupted so as to extend the length to which the signal can be transmitted over the same network. An important point to be noted about repeaters is that they do not amplify the signal. When the signal becomes weak, they copy the signal bit by bit and regenerate it at the original strength. It is a 2-port device.

Hub – A hub is basically a multiport repeater. A hub connects multiple wires coming from different branches, for example, the connector in star topology which connects different stations. Hubs cannot filter data, so data packets are sent to all connected devices. In other words, the collision domain of all hosts connected through Hub remains one. Also, they do not have the intelligence to find out the best path for data packets which leads to inefficiencies and wastage.

Link Layer Devices:

Bridge – A bridge operates at the data link layer. A bridge is a repeater, with add on the functionality of filtering content by reading the MAC addresses of source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2-port device.

Switch – A switch is a multiport bridge with a buffer and a design that can boost its efficiency (a large number of ports imply less traffic) and performance. A switch is a data link layer device. The switch can perform error checking before forwarding data, which makes it very efficient as it does not forward packets that have errors and forward good packets selectively to the correct port only. In other words, the switch divides the collision domain of hosts, but broadcast domain remains the same.

NIC – NIC or network interface card is a network adapter that is used to connect the computer to the network. It is installed in the computer to establish a LAN. It has a unique id that is written on the chip, and it has a connector to connect the cable to it. The cable acts as an interface between the computer and router or modem. NIC card is a layer 2 device which means that it works on both physical and data link layer of the network model.

Network layer Devices:

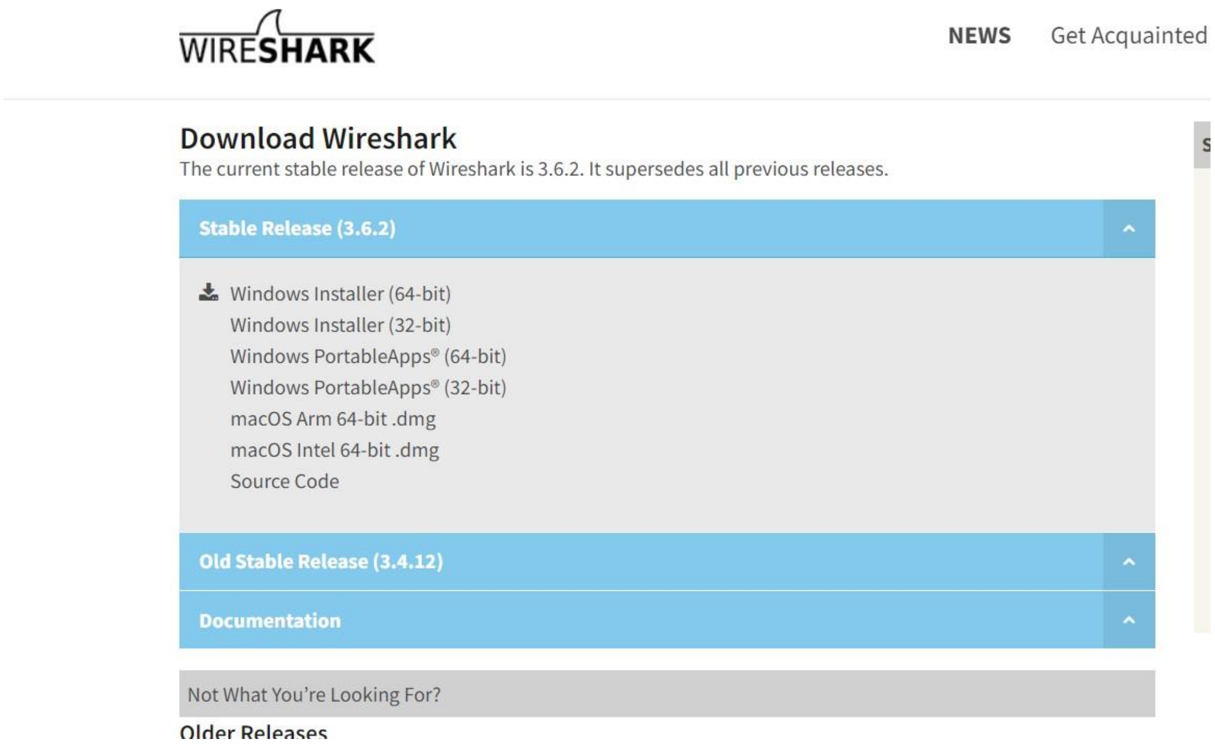
Routers – A router is a device like a switch that routes data packets based on their IP addresses. The router is mainly a Network Layer device. Routers normally connect LANs and WANs together and have a dynamically updating routing table based on which they make decisions on routing the data packets. Router divide broadcast domains of hosts connected through it.

Gateway – A gateway, as the name suggests, is a passage to connect two networks together that may work upon different networking models. They basically work as the messenger agents that take data from one system, interpret it, and transfer it to another system. Gateways are also called protocol converters and can operate at any network layer. Gateways are generally more complex than switches or routers.

EXPERIMENT - 2

Install and run a network diagnosis tool such as Tcp dump or Wireshark. Start capturing packets on an active interface, open a browser and type the address of your favourite search engine. Wait till the page loads and stop capture. List out the type and number of each type of packets captured.

INSTALLING WIRESHARK:



NEWS Get Acquainted

Download Wireshark

The current stable release of Wireshark is 3.6.2. It supersedes all previous releases.

Stable Release (3.6.2)

- Windows Installer (64-bit)
- Windows Installer (32-bit)
- Windows PortableApps® (64-bit)
- Windows PortableApps® (32-bit)
- macOS Arm 64-bit .dmg
- macOS Intel 64-bit .dmg
- Source Code

Old Stable Release (3.4.12)

Documentation

Not What You're Looking For?

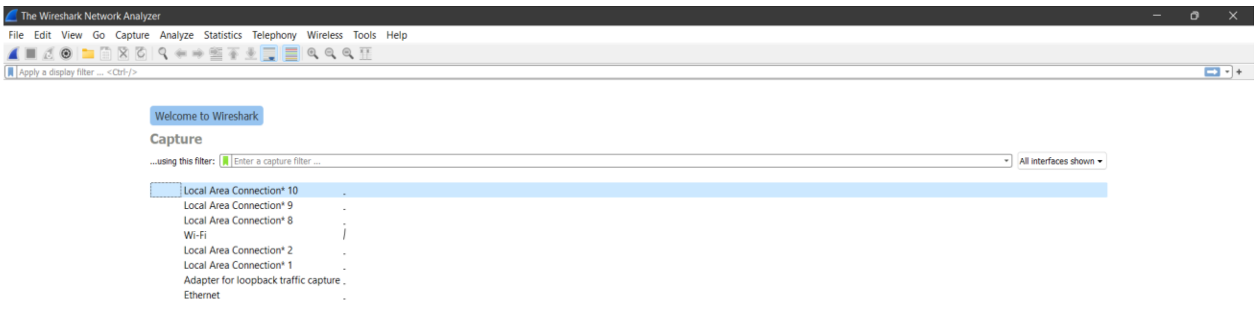
Older Releases

Go to: https://www.wireshark.org/download.html

Open the Wireshark after installing it.

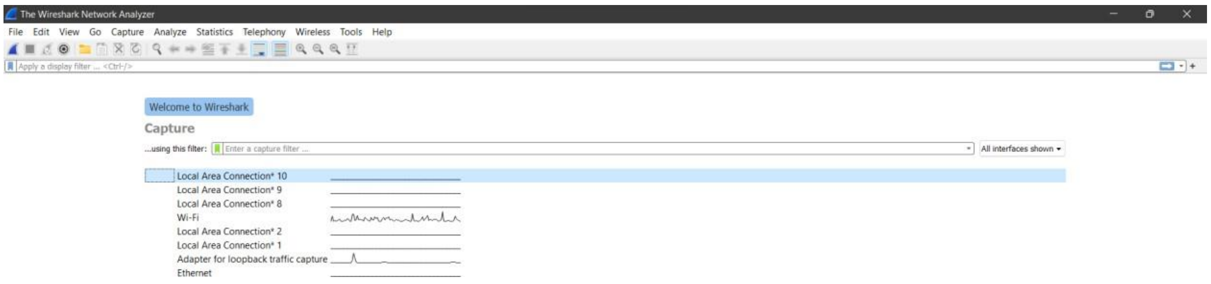
"WIFI":

BEFORE CONNECTION:

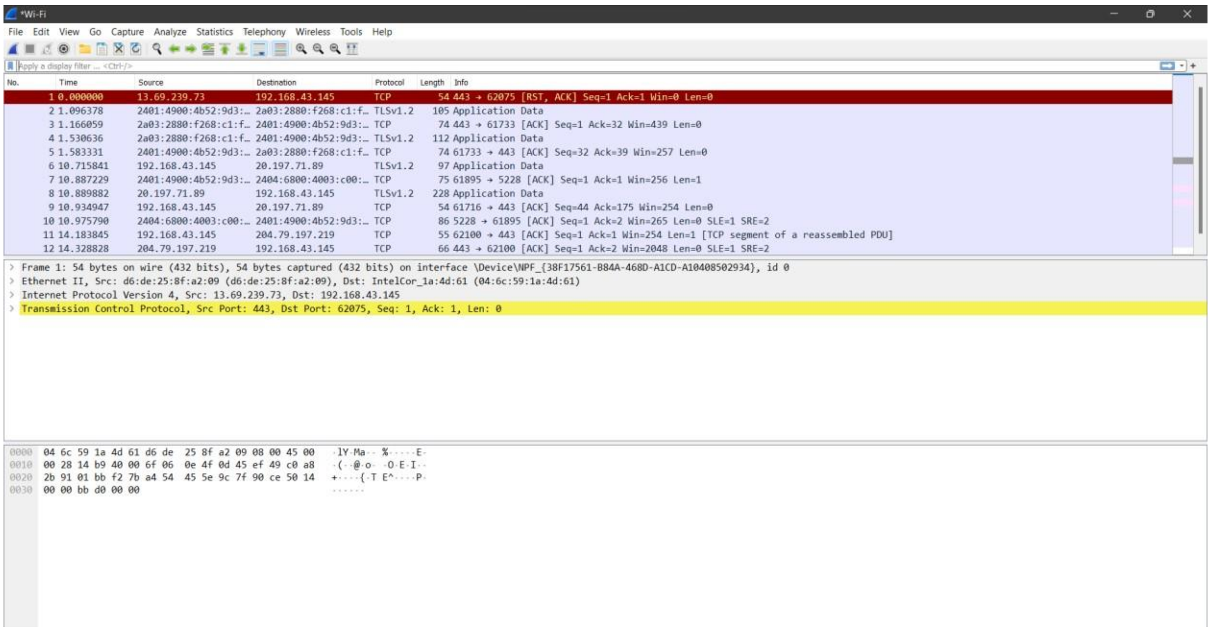

The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown

Local Area Connection* 10
Local Area Connection* 9
Local Area Connection* 8
Wi-Fi
Local Area Connection* 2
Local Area Connection* 1
Adapter for loopback traffic capture
Ethernet

AFTER CONNECTION:


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
Local Area Connection* 10
Local Area Connection* 9
Local Area Connection* 8
Wi-Fi
Local Area Connection* 2
Local Area Connection* 1
Adapter for loopback traffic capture
Ethernet

CAPTURING PACKETS:


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time        Source                 Destination            Protocol  Length Info
    1 0.000000    13.69.239.73           192.168.43.145         TCP         54 443 → 62075 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
    2 1.096378    2401:4900:4b52:9d3:…   2a03:2880:f268:c1:f…   TLSv1.2    105 Application Data
    3 1.166059    2a03:2880:f268:c1:f…   2401:4900:4b52:9d3:…   TCP         74 443 → 61733 [ACK] Seq=1 Ack=32 Win=439 Len=0
    4 1.530636    2a03:2880:f268:c1:f…   2401:4900:4b52:9d3:…   TLSv1.2    112 Application Data
    5 1.583331    2401:4900:4b52:9d3:…   2a03:2880:f268:c1:f…   TCP         74 61733 → 443 [ACK] Seq=32 Ack=39 Win=257 Len=0
    6 10.715841   192.168.43.145         20.197.71.89           TLSv1.2     97 Application Data
    7 10.887229   2401:4900:4b52:9d3:…   2404:6800:4003:c00:…   TCP         75 61895 → 5228 [ACK] Seq=1 Ack=1 Win=256 Len=1
    8 10.889882   20.197.71.89           192.168.43.145         TLSv1.2    228 Application Data
    9 10.934947   192.168.43.145         20.197.71.89           TCP         54 61716 → 443 [ACK] Seq=44 Ack=175 Win=254 Len=0
   10 10.975790   2404:6800:4003:c00:…   2401:4900:4b52:9d3:…   TCP         86 5228 → 61895 [ACK] Seq=1 Ack=2 Win=265 Len=0 SLE=1 SRE=2
   11 14.183845   192.168.43.145         204.79.197.219         TCP         55 62100 → 443 [ACK] Seq=1 Ack=1 Win=254 Len=1 [TCP segment of a reassembled PDU]
   12 14.328828   204.79.197.219         192.168.43.145         TCP         66 443 → 62100 [ACK] Seq=1 Ack=2 Win=2048 Len=0 SLE=1 SRE=2
```

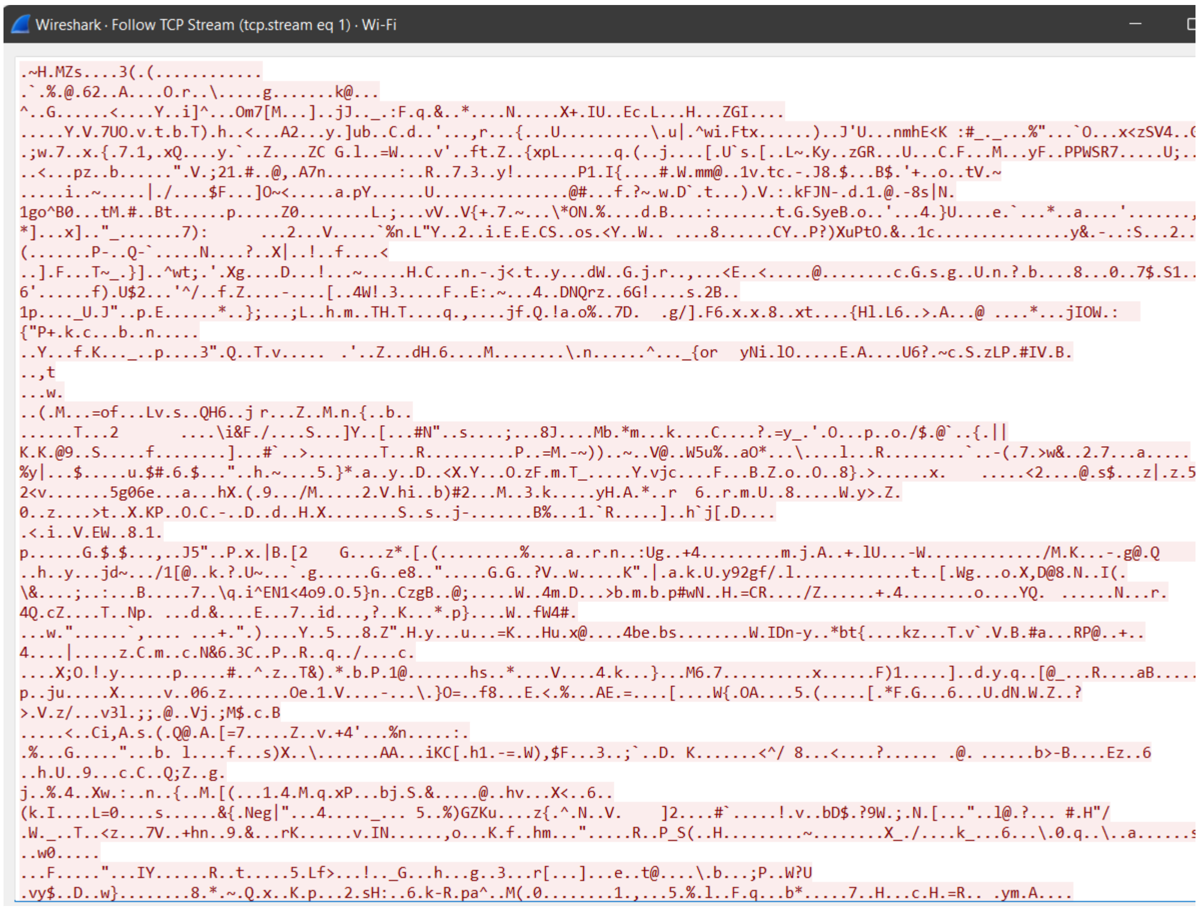
```
> Frame 1: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{38F17561-B84A-468D-A1CD-A10408502934}, id 0
> Ethernet II, Src: d6:de:25:8f:a2:09 (d6:de:25:8f:a2:09), Dst: IntelCor_1a:4d:61 (04:6c:59:1a:4d:61)
> Internet Protocol Version 4, Src: 13.69.239.73, Dst: 192.168.43.145
> Transmission Control Protocol, Src Port: 443, Dst Port: 62075, Seq: 1, Ack: 1, Len: 0
```

```
0000  04 6c 59 1a 4d 61 d6 de  25 8f a2 09 08 00 45 00   ·lY·Ma·· %·····E·
0010  00 28 14 b9 40 00 6f 06  0e 4f 0d 45 ef 49 c0 a8   ·(··@·o· ·O·E·I··
0020  2b 91 01 bb f2 7b a4 54  45 5e 9c 7f 90 ce 50 14   +····{·T E^····P·
0030  00 00 bb d0 00 00                                  ······
```

Type of packets captured.

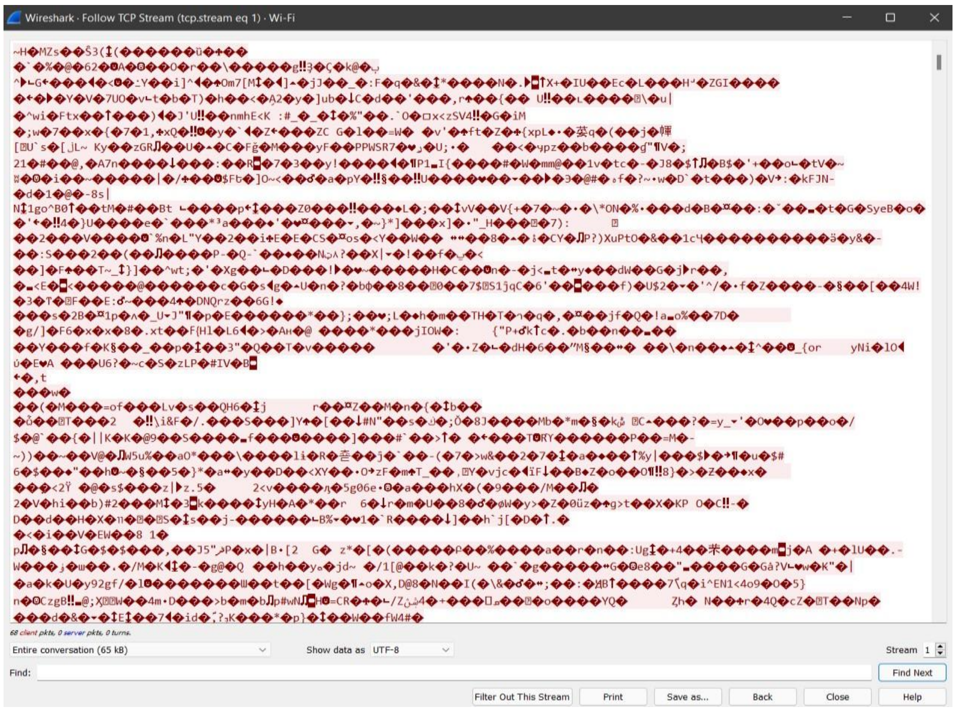
1. TCP:

a) ASCII CODE:


Wireshark · Follow TCP Stream (tcp.stream eq 1) · Wi-Fi

```
.~H.MZs....3(.(............
.`.%.@.62..A....O.r..\.....g.......k@...
^..G......<....Y..i]^...Om7[M...]..jJ.._.:F.q.&..*....N.....X+.IU..Ec.L...H...ZGI....
.....Y.V.7UO.v.t.b.T).h..<...A2...y.]ub..C.d..'...,r...{...U..........\.u|.^wi.Ftx......)..J'U...nmhE<K :#_._...%"...`O...x<zSV4..
.;w.7..x.{.7.1,.xQ....y.`..Z....ZC G.l..=W....v'..ft.Z..{xpL......q.(..j....[.U`s.[..L~.Ky..zGR...U...C.F...M...yF..PPWSR7.....U;..
..<...pz..b......".V.;21.#..@,.A7n........:..R..7.3..y!.......P1.I{....#.W.mm@..1v.tc.-.J8.$...B$.'+..o..tV.~
.....i..~.....|./....$F...]O~<.....a.pY......U...............@#...f.?~.w.D`.t...).V.:.kFJN-.d.1.@.-8s|N.
1go^B0...tM.#..Bt......p.....Z0........L.;...vV..V{+.7.~...\*ON.%....d.B....:.......t.G.SyeB.o..'...4.}U....e.`...*..a....'.......,
*]...x].."_.......7):      ...2...V.....`%n.L"Y..2..i.E.E.CS..os.<Y..W.. ....8......CY..P?)XuPtO.&..1c...............y&.-..:S...2..
(.......P-..Q-`.....N....?..X|..!..f....<
..].F...T~_.}]..^wt;.'.Xg....D...!...~.....H.C...n.-.j<.t..y...dW..G.j.r..,...<E..<.....@........c.G.s.g..U.n.?.b....8...0..7$.S1..
6'......f).U$2...'^/..f.Z....-....[..4W!.3.....F..E:.~...4..DNQrz..6G!....s.2B..
1p...._U.J"..p.E......*..};...;L..h.m..TH.T....q.,....jf.Q.!a.o%..7D.  .g/].F6.x.x.8..xt....{Hl.L6..>.A...@ ....*...jIOW.:
{"P+.k.c...b..n.....
..Y...f.K..._..p....3".Q..T.v.....  .'..Z...dH.6....M........\.n......^..._{or  yNi.lO.....E.A....U6?.~c.S.zLP.#IV.B.
..,t
...w.
..(.M...=of...Lv.s..QH6..j r...Z..M.n.{..b..
......T...2        ....\i&F./....S...]Y..[...#N"..s....;...8J....Mb.*m...k....C....?.=y_.'.O...p..o./$.@`..{.||
K.K.@9..S.....f........]...#`..>........T...R..........P..=M.-~))..~..V@..W5u%..aO*...\....l...R.........`..-(.7.>w&..2.7...a.....
%y|...$.....u.$#.6.$..."..h.~....5.}*.a..y..D..<X.Y...O.zF.m.T_.....Y.vjc....F...B.Z.o..O..8}.>......x.    .....<2....@.s$...z|.z.5
2<v.......5g06e...a...hX.(.9.../M.....2.V.hi..b)#2...M..3.k.....yH.A.*..r  6..r.m.U..8.....W.y>.Z.
0..z....>t..X.KP..O.C.-..D..d..H.X........S..s..j-.......B%...1.`R.....]..h`j[.D....
.<.i..V.EW..8.1.
p......G.$.$...,..J5"..P.x.|B.[2    G....z*.[.(.........%....a..r.n..:Ug..+4.........m.j.A..+.lU...-W............./M.K...-.g@.Q
..h..y...jd~.../1[@..k.?.U~...`.g......G..e8.."......G.G..?V..w.....K".|.a.k.U.y92gf/.l.............t..[.Wg...o.X,D@8.N..I(.
\&....;..:...B.....7..\q.i^EN1<4o9.O.5}n..CzgB..@;.....W..4m.D...>b.m.b.p#wN..H.=CR..../Z......+.4........o....YQ.  ......N...r.
4Q.cZ....T..Np. ...d.&....E...7..id...,?..K...*.p}....W..fW4#.
...w."......`,.... ...+.".)....Y..5...8.Z".H.y...u...=K...Hu.x@....4be.bs........W.IDn-y..*bt{....kz...T.v`.V.B.#a...RP@..+..
4....|.....z.C.m..c.N&6.3C..P..R..q../....c.
....X;O.!.y......p.....#..^.z..T&).*.b.P.1@.......hs..*....V....4.k...}...M6.7..........x......F)1.....]..d.y.q..[@_...R.....aB.....
p..ju.....X.....v..06.z.......Oe.1.V....-...\.}O=..f8...E.<.%...AE.=....[....W{.OA....5.(.....[.*F.G...6...U.dN.W.Z..?
>.V.z/...v3l.;;.@..Vj.;M$.c.B
.....<..Ci,A.s.(.Q@.A.[=7.....Z..v.+4'...%n.....:.
.%...G....."...b. l....f...s)X..\.......AA...iKC[.h1.-=.W),$F...3..;`..D. K.......<^/ 8...<....?...... .@. ......b>-B....Ez..6
..h.U..9...c.C..Q;Z..g.
j..%.4..Xw.:..n..{..M.[(...1.4.M.q.xP...bj.S.&.....@..hv...X<..6..
(k.I....L=0....s......&{.Neg|"...4....._... 5..%)GZKu....z{.^.N..V.     ]2....#`.....!.v..bD$.?9W.;.N.[...."..l@.?... #.H"/
.W._..T..<z...7V..+hn..9.&...rK......v.IN......,o...K.f..hm..."......R..P_S(..H.........~........X_./....k_...6...\.0.q..\..a......
..w0.....
...F....."...IY......R..t.....5.Lf>...!.._G...h...g..3...r[...]...e..t@....\.b...;P..W?U
.vy$..D..w}........8.*.~.Q.x..K.p...2.sH:..6.k-R.pa^..M(.0........1.,...5.%.l..F.q...b*.....7..H...c.H.=R.. .ym.A....
```

b) UTF-8:


Wireshark · Follow TCP Stream (tcp.stream eq 1) · Wi-Fi

```
~H�MZs��$3(↨(�������ü�✝��
�`�%�@�62��A������O�r��\������g‼ȝ�Ç�k@�ټ
^▶∟G←����◀�<��∸Y��i]^◀�↑Om7[M↨�◀]▲�jJ��_�:F�q�&�↨*����N�.▶▀↑X+�IU��Ec�L���H˩�ZGI����
�←�▶�Y�V�7UO�v∟t�b�T)�h��<�A2�y�]ub�↓C�d��'���,r↑��{�� U‼��ʟ�����\�u|
�^wi�Ftx��↑���)◀�J'U‼��nmhE<K :#_�_�↨�%"��.`O�□x<zSV4‼�G�iM
�;w�7��x�{�7�1,♣xQ�‼��y�`◀�Z←���ZC G�l��=W� �v'�↑ft�Z�♣{xpL♦•�葵q�(��j�幝
[U`s�[ﺎL~ Ky��zGR♫��U�▲�C�Fğ�M���yF��PPWSR7�♥ﹷ�U;•�    ��<�ɥpz��b�����ɗ"¶V�;
21�#��@,�A7n����↓���:��R▀�7�3��y!����◀�¶P1▬I{����#�W�mm@��1v�tc�-�J8�$↑♫�B$�'+��o∟�tV�~
��i��~�����|�/♣���$Ft�]O~<���♂�a�pY�‼§��‼U������♥��▾��▶�Э�@#�ₒf�?~•w�D`�t���)�V➔:�kFJN-
�d�1�@�-8s|
N↨1go^B0↑��tM�#��Bt ∟����p←↨���Z0���‼����♦L�;��↨vV��V{+�7�~�•�\*ON�%•���d�B�¤��:�˘��▬�t�G�SyeB�o�
�'←�‼4�}U�����e�`���*³a����♦'�♥¤���▾,�~}*]���x]�•"_H������7):
��2���V�����`%n�L"Y��2��i♣E�E�CS�¤os�<Y��W�� ↔��8�▲�˚�CY�♫P?)XuPtO�&��1cЧ������������ä�y&�-
��:S���2��(��♫�����P-�Q-`����♦Nↄٯ?��X|▾�!��f���<
��]�F↑��T~_↨}]��^wt;�'�Xg��∟�D���!▶�♥~������H�C��n�-�j<▬t�↔y♦���dW���G�j▶r��,
�▬<E�▀<�������@�������c�G�s◀g�▲U�n�?�bɸ��8���0��7$S1ĵqC�6'��▀����f)�U$2�▾�'^/�•f�Z����-�§��[��4W!
�3�T��F��E:♂~���4↑�DNQrz��6G!♦
���s�2B�¤1p�٨�_U▾J"¶�p�E�������*��};��♥;L�♦h�m��TH�T�˥�q�,�¤��jf�Q�!a▬o%��7D�
�g/]�F6�x�x�8�.xt��F{Hl�L6◀�>�AH�@ ����*���jIOW�:    {"P+♂k↑c�.�b��n��▬��
��Y���f�K§��_��p�↨��3"�Q��T�v�����       �'�•Z�∟�dH�6��”M§��↔� ��\�n���♦▲�↨^��_{or    yNi�1O◀
ú�E♥A ���U6?�~c�S�zLP�#IV�B▀
←�,t
���w�
��(�M���=of���Lv�s��QH6�↨j      r��¤Z��M�n�{�↨b��
�ǒ���T���2   �‼\i&F�/.���S���]Y↑�[��↓#N"��s�ﻝ�;Ŏ�8J����Mb�*m�§�kﺶ C▲���?�=y_▾'�O♥���p���o�/
$�@`��{�||K�K�@9���S����▬f���������]���#`��>↑� �←���TRY��������P��=M�-
~))��~��V@�♫W5u%��aO*���\����1i�R�춘��j�`��-(�7�>w&��2�7�↨�a♦���↑%y|���$▶�➔¶�u�$#
6�$���♦"��h�~�§��5�}*�a↔�y��D��<XY��•O➔zF�m↑T_��,Y�vjc�◀ïF↓��B♦Z�o���O¶‼8}�>�Z���♦x�
���<2Ÿ �@�s$���z|▶z.5�     2<v�����л�5g06e•��a���hX�(�9���/M��♫�
2�V�hi��b)#2���M↨�3▀k����↨yH�A�*��r  6�↓r�m�U��8�♂�øW�y>�Z�0üz�↑g>t��X�KP O�C‼-�
D��d��H�X�Π���S�↨s��j-������∟B%▾�♥1�`R����↓]��h`j[�D�↑.�
�<�i��V�EW��8 1�
p♫�§��↨G�$�$���,��J5"ﮃP�x�|B•[2  G� z*�[�(������Ρ��%�����a��r�n��:Ug↨�+4��米�����m▀j�A �+�1U��.-
W����ﺯ�ш��.�/M�K◀↨�-�g@�Q  ��h��yₐ�jd~ �/1[@��k�?�U~ ��`�g������↔G��e8��"▬����G�Gå?V∟♥w�K"�|
�a�k�U�y92gf/�1���������Ш��t��[�Wg�¶▲o�X,D@8�N��I(�\&��♂�↔;��:�ӸB↑�����7ʕq�i^EN1<4o9�O�5}
n��CzgB‼▬@;X��W��4m•D���>b�m�b♫p#wN♫▀H�=CR�✝�∟/Zﺷ4�+���▯ₐ����o�����YQ�      Zh� N��✝r�4Q�cZ��T��Np�
���d�&�▾�↨E↨��7◀�id�ٛ?ₛK���*�p}�↨��W��fW4#�
```

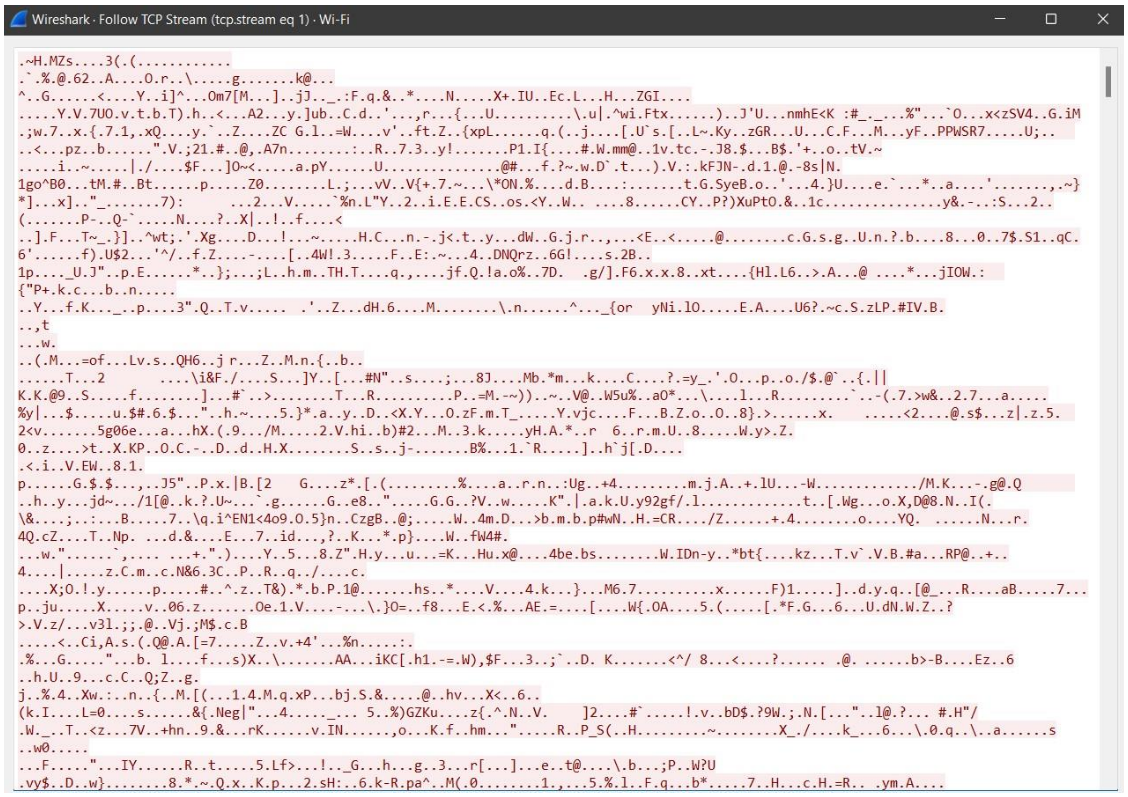
68 client pkts, 0 server pkts, 0 turns.

Entire conversation (65 kB) | Show data as UTF-8 | Stream 1

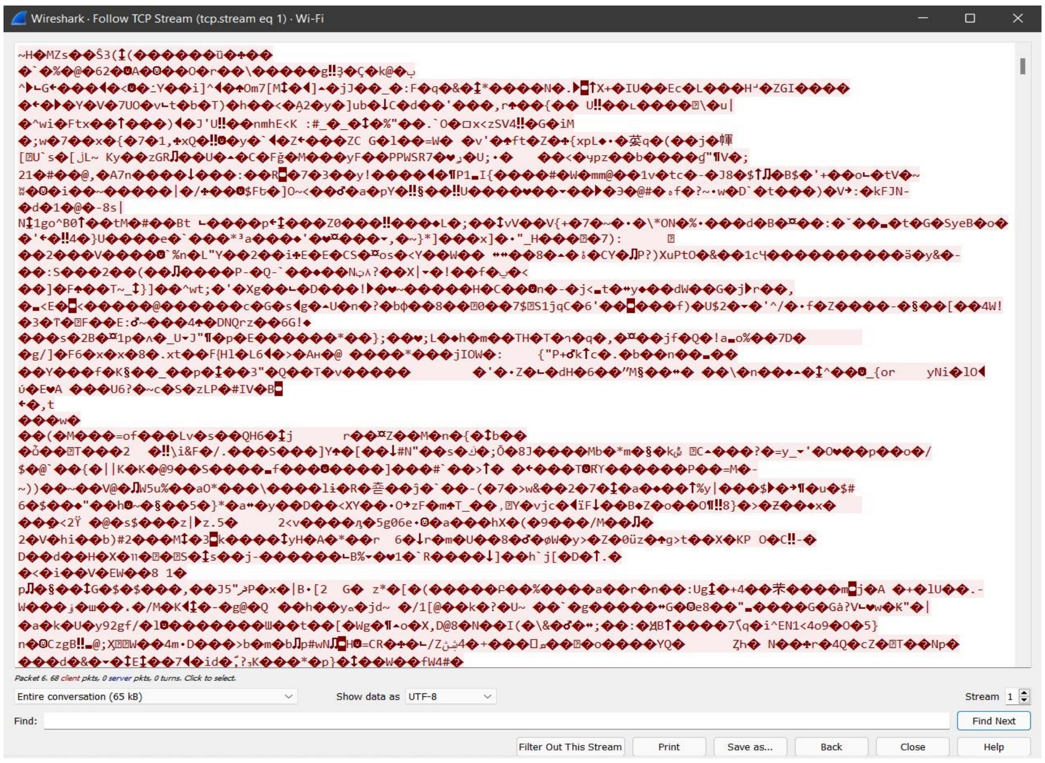
Find:

c) SSL (through TCP stream):

ASCII:


Wireshark · Follow TCP Stream (tcp.stream eq 1) · Wi-Fi

```
.~H.MZs....3(.(............
.`.%.@.62..A....O.r..\.....g.......k@...
^..G......<....Y..i]^...Om7[M...]..jJ.._.:F.q.&..*....N.....X+.IU..Ec.L...H...ZGI....
.....Y.V.7UO.v.t.b.T).h..<...A2...y.]ub..C.d..'...,r...{...U.........\.u|.^wi.Ftx......)..J'U...nmhE<K :#_._...%"...`O...x<zSV4..G.iM
.;w.7..x.{.7.1,.xQ....y.`..Z....ZC G.l..=W....v'..ft.Z..{xpL......q.(..j....[.U`s.[..L~.Ky..zGR...U...C.F...M...yF..PPWSR7.....U;..
..<...pz..b......".V.;21.#..@,.A7n........:..R..7.3..y!.......P1.I{....#.W.mm@..1v.tc.-.J8.$...B$.'+..o..tV.~
.....i..~.....|./....$F...]O~<.....a.pY......U...............@#...f.?~.w.D`.t...).V.:.kFJN-.d.1.@.-8s|N.
1go^B0...tM.#..Bt......p.....Z0........L.;...vV..V{+.7.~...\*ON.%....d.B....:.......t.G.SyeB.o..'...4.}U....e.`...*..a....'.......,.~}
*]...x].."_.......7):      ...2...V.....`%n.L"Y..2..i.E.E.CS..os.<Y..W.. ....8......CY..P?)XuPtO.&..1c..............y&.-..:S...2..
(.......P-..Q-`.....N....?..X|..!..f....<
..].F...T~_.}]..^wt;.'.Xg....D...!...~.....H.C...n.-.j<.t..y...dW..G.j.r..,...<E..<.....@........c.G.s.g..U.n.?.b....8...0..7$.S1..qC.
6'......f).U$2...'^/..f.Z....-....[..4W!.3.....F..E:.~...4..DNQrz..6G!....s.2B..
1p...._U.J"..p.E......*..};...;L..h.m..TH.T....q.,....jf.Q.!a.o%..7D.  .g/].F6.x.x.8..xt....{Hl.L6..>.A...@ ....*...jIOW.:
{"P+.k.c...b..n.....
..Y...f.K..._..p....3".Q..T.v.....  .'..Z...dH.6....M........\.n......^..._{or   yNi.lO.....E.A....U6?.~c.S.zLP.#IV.B.
..,t
...w.
..(.M...=of...Lv.s..QH6..j r...Z..M.n.{..b..
......T...2       ....\i&F./....S...]Y..[...#N"..s....;...8J....Mb.*m...k....C....?.=y_.'.O...p..o./$.@`..{.||
K.K.@9..S.....f........]...#`..>........T...R..........P..=M.-~))..~..V@..W5u%..aO*...\....l...R.........`..-(.7.>w&..2.7...a.....
%y|...$.....u.$#.6.$...."..h.~....5.}*.a..y..D..<X.Y...O.zF.m.T_.....Y.vjc....F...B.Z.o..O..8}.>......x.    .....<2....@.s$...z|.z.5.
2<v.......5g06e...a...hX.(.9.../M.....2.V.hi..b)#2...M..3.k.....yH.A.*..r  6..r.m.U..8.....W.y>.Z.
0..z....>t..X.KP..O.C.-..D..d..H.X........S..s..j-.......B%...1.`R.....]..h`j[.D....
.<.i..V.EW..8.1.
p......G.$.$...,..J5"..P.x.|B.[2    G....z*.[.(.........%....a..r.n..:Ug..+4.........m.j.A..+.1U...-W............./M.K...-.g@.Q
..h..y...jd~.../1[@..k.?.U~...`.g......G..e8..".....G.G..?V..w.....K".|.a.k.U.y92gf/.l.............t..[.Wg...o.X,D@8.N..I(.
\&....;..:...B.....7..\q.i^EN1<4o9.O.5}n..CzgB..@;.....W..4m.D...>b.m.b.p#wN..H.=CR..../Z......+.4........o....YQ.  ......N...r.
4Q.cZ....T..Np. ...d.&....E...7..id...,?..K...*.p}....W..fW4#.
...w."......`,.... ...+.".)....Y..5...8.Z".H.y...u...=K...Hu.x@....4be.bs........W.IDn-y..*bt{....kz...T.v`.V.B.#a...RP@..+..
4....|.....z.C.m..c.N&6.3C..P..R..q../....c.
....X;O.!.y......p.....#..^.z..T&).*.b.P.1@.......hs..*....V....4.k...}...M6.7..........x......F)1.....]..d.y.q..[@_...R....aB.....7...
p..ju.....X.....v..06.z.......Oe.1.V....-...\.}O=..f8...E.<.%...AE.=....[....W{.OA....5.(.....[.*F.G...6...U.dN.W.Z..?
>.V.z/...v3l.;;.@..Vj.;M$.c.B
.....<..Ci,A.s.(.Q@.A.[=7.....Z..v.+4'...%n.....:.
.%...G....."...b. l....f...s)X..\.......AA...iKC[.h1.-=.W),$F...3..;`..D. K.......<^/ 8...<....?...... .@. ......b>-B....Ez..6
..h.U..9...c.C..Q;Z..g.
j..%.4..Xw.:..n..{..M.[(...1.4.M.q.xP...bj.S.&.....@..hv...X<..6..
(k.I....L=0....s......&{.Neg|"...4....._... 5..%)GZKu....z{.^.N..V.     ]2....#`.....!.v..bD$.?9W.;.N.[..."..1@.?... #.H"/
.W._..T..<z...7V..+hn..9.&...rK......v.IN......,o...K.f..hm...".....R..P_S(..H.........~........X_./....k_...6...\.0.q..\..a......s
..w0.....
...F....."...IY......R..t.....5.Lf>...!.._G...h...g..3...r[...]...e..t@....\.b...;P..W?U
.vy$..D..w}........8.*.~.Q.x..K.p...2.sH:..6.k-R.pa^..M(.0........1.,...5.%.l..F.q...b*.....7..H...c.H.=R.. .ym.A....
```



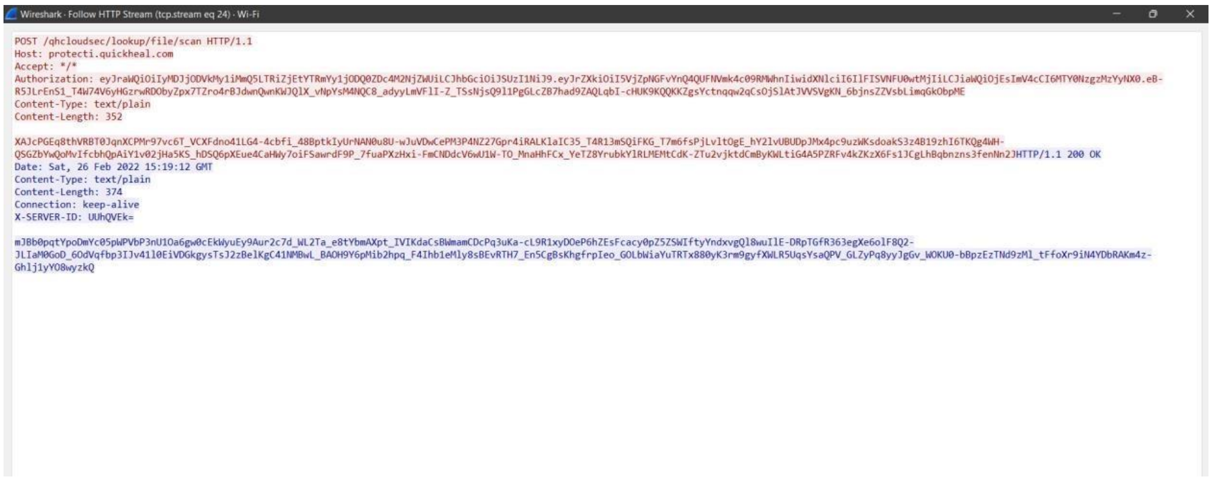
```
[illegible]
```

Packet 6. 68 client pkts, 0 server pkts, 0 turns. Click to select.

Entire conversation (65 kB) · Show data as UTF-8 · Stream 1

2.<u>HTTP: (through HTTP stream)</u>

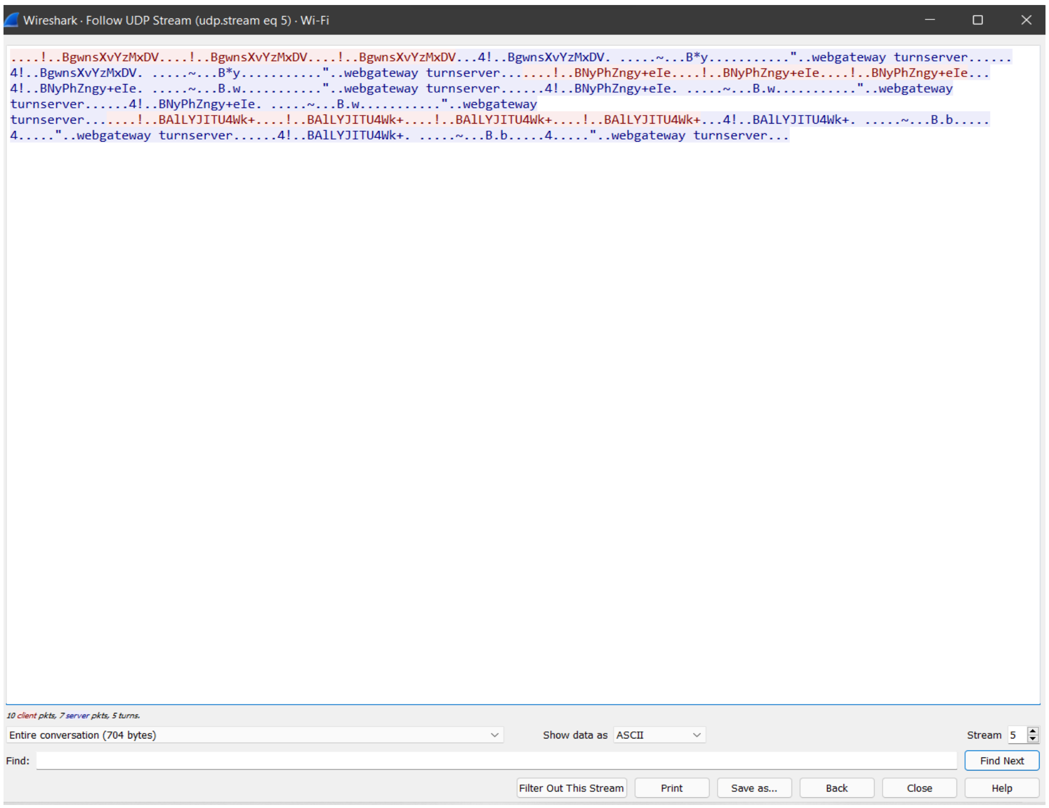
<u>ASCII:</u>



```
POST /qhcloudsec/lookup/file/scan HTTP/1.1
Host: protecti.quickheal.com
Accept: */*
Authorization: eyJraWQiOiIyMDJjODVkMy1iMmQ5LTRiZjEtYTRmYy1jODQ0ZDc4M2NjZWUiLCJhbGciOiJSUzI1NiJ9.eyJrZXkiOiI5VjZpNGFvYnQ4QUFNVmk4c09RMWhnIiwidXNlciI6IlFISVNFU0wtMjIiLCJiaWQiOjEsImV4cCI6MTY0NzgzMzYyNX0.eB-
R5JLrEnS1_T4W74V6yHGzrwRDObyZpx7TZro4rBJdwnQwnKWJQlX_vNpYsM4NQC8_adyyLmVFlI-Z_TSsNjsQ9l1PgGLcZB7had9ZAQLqbI-cHUK9KQQKKZgsYctnqqw2qCsOjSlAtJVVSVgKN_6bjnsZZVsbLimqGkObpME
Content-Type: text/plain
Content-Length: 352

XAJcPGEq8thVRBT0JqnXCPMr97vc6T_VCXFdno41LG4-4cbfi_48BptkIyUrNAN0u8U-wJuVDwCePM3P4NZ27Gpr4iRALKlaIC35_T4R13mSQiFKG_T7m6fsPjLvltOgE_hY2lvUBUDpJMx4pc9uzWKsdoakS3z4B19zhI6TKQg4WH-
QSGZbYwQoMvIfcbhQpAiY1v02jHa5KS_hDSQ6pXEue4CaHWy7oiFSawrdF9P_7fuaPXzHxi-FmCNDdcV6wU1W-TO_MnaHhFCx_YeTZ8YrubkYlRLMEMtCdK-ZTu2vjktdCmByKWLtiG4A5PZRFv4kZKzX6Fs1JCgLhBqbnzns3fenNn2JHTTP/1.1 200 OK
Date: Sat, 26 Feb 2022 15:19:12 GMT
Content-Type: text/plain
Content-Length: 374
Connection: keep-alive
X-SERVER-ID: UUhQVEk=

mJBb0pqtYpoDmYc05pWPVbP3nU1Oa6gw0cEkWyuEy9Aur2c7d_WL2Ta_e8tYbmAXpt_IVIKdaCsBWmamCDcPq3uKa-cL9R1xyDOeP6hZEsFcacy0pZ5ZSWIftyYndxvgQl8wuIlE-DRpTGfR363egXe6olF8Q2-
JLIaM0GoD_6OdVqfbp3IJv41l0EiVDGkgysTsJ2zBelKgC41NMBwL_BAOH9Y6pMib2hpq_F4Ihb1eMly8sBEvRTH7_En5CgBsKhgfrpIeo_GOLbWiaYuTRTx880yK3rm9gyfXWLR5UqsYsaQPV_GLZyPq8yyJgGv_WOKU0-bBpzEzTNd9zMl_tFfoXr9iN4YDbRAKm4z-
Ghlj1yYO8wyzkQ
```

3. STUN: (through UDP Stream)

a) ASCII:

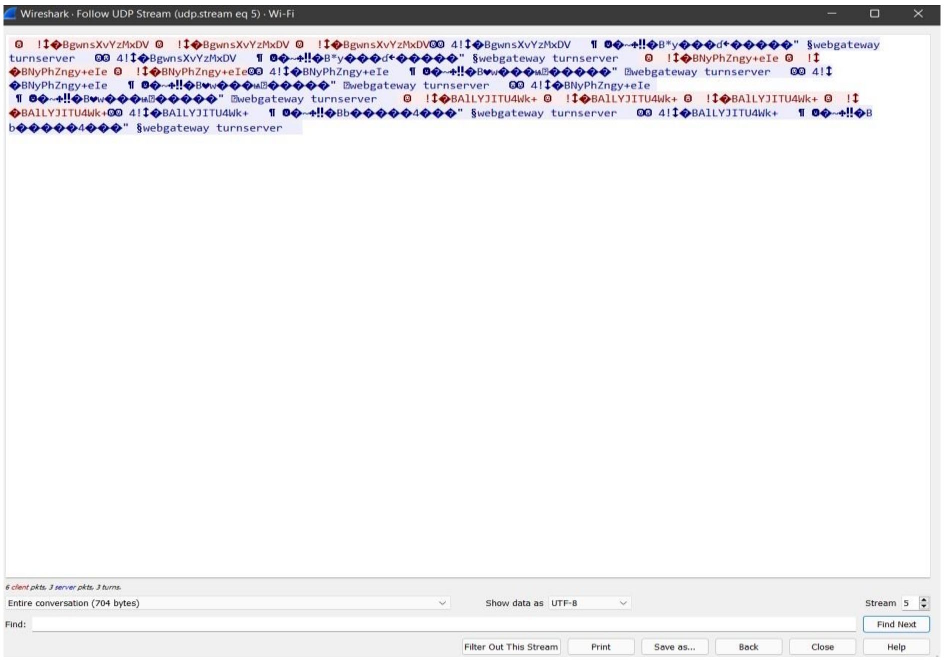


```
....!..BgwnsXvYzMxDV....!..BgwnsXvYzMxDV....!..BgwnsXvYzMxDV...4!..BgwnsXvYzMxDV. .....~...B*y..........."..webgateway turnserver......
4!..BgwnsXvYzMxDV. .....~...B*y..........."..webgateway turnserver.......!..BNyPhZngy+eIe....!..BNyPhZngy+eIe....!..BNyPhZngy+eIe...
4!..BNyPhZngy+eIe. .....~...B.w..........."..webgateway turnserver......4!..BNyPhZngy+eIe. .....~...B.w..........."..webgateway
turnserver......4!..BNyPhZngy+eIe. .....~...B.w..........."..webgateway
turnserver.......!..BAlLYJITU4Wk+....!..BAlLYJITU4Wk+....!..BAlLYJITU4Wk+....!..BAlLYJITU4Wk+...4!..BAlLYJITU4Wk+. .....~...B.b.....
4....."..webgateway turnserver......4!..BAlLYJITU4Wk+. .....~...B.b.....4....."..webgateway turnserver...
```

10 client pkts, 7 server pkts, 5 turns.

Entire conversation (704 bytes) Show data as ASCII Stream 5

b) UTF-8:



```
☺  !↕�BgwnsXvYzMxDV ☺  !↕�BgwnsXvYzMxDV ☺  !↕�BgwnsXvYzMxDV☺☺ 4!↕�BgwnsXvYzMxDV   ¶ ☺�~→‼�B*y���d←�����" §webgateway
turnserver   ☺☺ 4!↕�BgwnsXvYzMxDV   ¶ ☺�~→‼�B*y���d←�����" §webgateway turnserver    ☺  !↕�BNyPhZngy+eIe ☺  !↕
�BNyPhZngy+eIe ☺  !↕�BNyPhZngy+eIe☺☺ 4!↕�BNyPhZngy+eIe   ¶ ☺�~→‼�B♥w���␦�����" ␦webgateway turnserver   ☺☺ 4!↕
�BNyPhZngy+eIe   ¶ ☺�~→‼�B♥w���␦�����" ␦webgateway turnserver   ☺☺ 4!↕�BNyPhZngy+eIe
 ¶ ☺�~→‼�B♥w���␦�����" ␦webgateway turnserver    ☺  !↕�BAlLYJITU4Wk+ ☺  !↕�BAlLYJITU4Wk+ ☺  !↕�BAlLYJITU4Wk+ ☺  !↕
�BAlLYJITU4Wk+☺☺ 4!↕�BAlLYJITU4Wk+   ¶ ☺�~→‼�Bb������4���" §webgateway turnserver   ☺☺ 4!↕�BAlLYJITU4Wk+   ¶ ☺�~→‼�B
b������4���" §webgateway turnserver
```

6 client pkts, 3 server pkts, 3 turns.

Entire conversation (704 bytes) Show data as UTF-8 Stream 5

EXPERIMENT - 3

Write a program to create a server that listens to port 5003 using stream sockets. Write a simple client program to connect to the server. Send a simple text message “Hello” from the client to the server and the server to the client and close the connection.

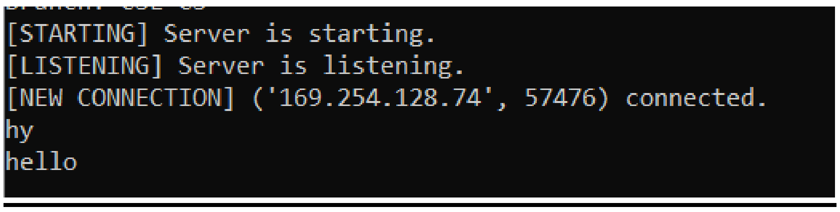
<u>Server code :</u>

```
print("This is experiment 3rd server code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
import socket
IP = socket.gethostbyname(socket.gethostname())
PORT = 5004
ADDR = (IP, PORT)
SIZE = 1024
FORMAT = "utf-8"
def main():
    print("[STARTING] Server is starting.")
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(ADDR)
    server.listen(2)
    print("[LISTENING] Server is listening.")
    while True:
        conn, addr = server.accept()
        print(f"[NEW CONNECTION] {addr} connected.")
        while True:
            msg = conn.recv(SIZE).decode(FORMAT)
            print(msg)
            if msg == "Bye" or msg == "bye" :
                conn.close()
                print(f"[DISCONNECTED] {addr} disconnected.")
                break

if __name__ == "__main__":
    main()
```

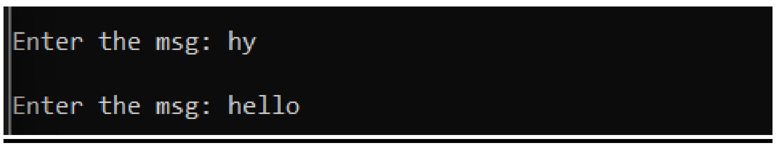
client code:

```python
print("This is experiment 3rd client code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
import socket
IP = socket.gethostbyname(socket.gethostname())
PORT = 5004
ADDR = (IP, PORT)
FORMAT = "utf-8"
SIZE = 1024
def main():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(ADDR)
    while True:
        msg = input("\nEnter the msg: ")

        client.send(msg.encode(FORMAT))
        if msg == "Bye" or msg == "bye" :
            break
    client.close()

if __name__ == "__main__":
    main()
```

OUTPUT:



```
[STARTING] Server is starting.
[LISTENING] Server is listening.
[NEW CONNECTION] ('169.254.128.74', 57476) connected.
hy
hello
```



```
Enter the msg: hy

Enter the msg: hello
```

EXPERIMENT - 4

Write a program to create a chat server that listens to port 5004 using stream sockets. Write a simple client program to connect to the server. Send multiple text messages from the client to the server and vice versa. When either party types "Bye", close the connection.

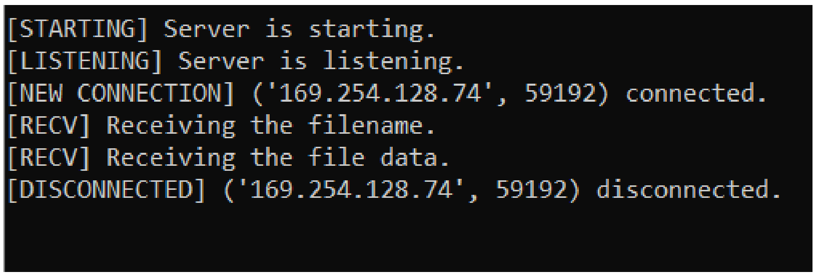
Server code :

```
print("This is experiment 4th server code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
import socket
IP = socket.gethostbyname(socket.gethostname())
PORT = 5005
ADDR = (IP, PORT)
SIZE = 2048
FORMAT = "utf-8"
def main():
    print("[STARTING] Server is starting.")
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(ADDR)
    server.listen()
    print("[LISTENING] Server is listening.")
    while True:
        conn, addr = server.accept()
        print(f"[NEW CONNECTION] {addr} connected.")
        filename = conn.recv(SIZE).decode(FORMAT)
        print(f"[RECV] Receiving the filename.")
        file = open(filename, "w")
        conn.send("Filename received.".encode(FORMAT))
        data = conn.recv(SIZE).decode(FORMAT)
        print(f"[RECV] Receiving the file data.")
        file.write(data)
        conn.send("File data received".encode(FORMAT))
        file.close()
        conn.close()
        print(f"[DISCONNECTED] {addr} disconnected.")
if __name__ == "__main__":
    main()
```

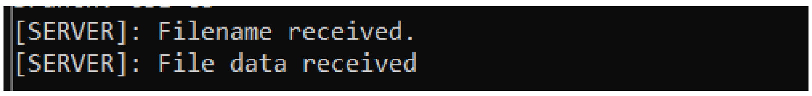
client code:

```
print("This is experiment 4th client code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")

import socket
IP = socket.gethostbyname(socket.gethostname())
PORT = 5005
ADDR = (IP, PORT)
FORMAT = "utf-8"
SIZE = 2048
def main():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(ADDR)
    file = open("data/textfile.txt", "r")
    data = file.read()
    client.send("yt.txt".encode(FORMAT))
    msg = client.recv(SIZE).decode(FORMAT)
    print(f"[SERVER]: {msg}")
    client.send(data.encode(FORMAT))
    msg = client.recv(SIZE).decode(FORMAT)
    print(f"[SERVER]: {msg}")
    file.close()
    client.close()
if __name__ == "__main__":
    main()
```

OUTPUT:



```
[STARTING] Server is starting.
[LISTENING] Server is listening.
[NEW CONNECTION] ('169.254.128.74', 59192) connected.
[RECV] Receiving the filename.
[RECV] Receiving the file data.
[DISCONNECTED] ('169.254.128.74', 59192) disconnected.
```



```
[SERVER]: Filename received.
[SERVER]: File data received
```

EXPERIMENT - 5

Write a program to create a server that listens to port 5005 using stream sockets. Write a simple client program to connect to the server. The client should request for a text file and the server should return the file before terminating the connection.

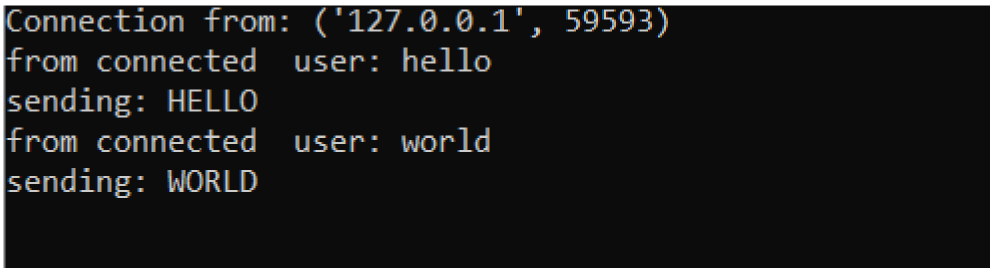
<u>Server code :</u>

```python
print("This is experiment 5th server code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
import socket
def Main():
    host = "127.0.0.1"
    port = 5000

    mySocket = socket.socket()
    mySocket.bind((host, port))

    mySocket.listen(1)
    conn, addr = mySocket.accept()
    print("Connection from: " + str(addr))
    while True:
        data = conn.recv(1024).decode()
        if not data:
            break
        print("from connected  user: " + str(data))

        data = str(data).upper()
        print("sending: " + str(data))
        conn.send(data.encode())

    conn.close()


if __name__ == '__main__':
    Main()
```

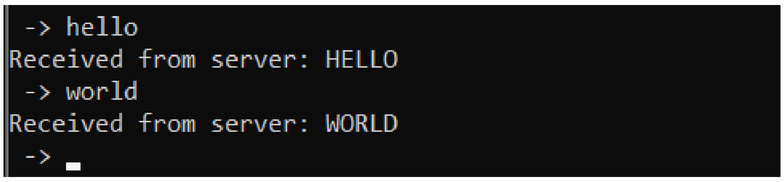
client code:

```python
print("This is experiment 5th client code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
import socket

def Main():
    host = '127.0.0.1'
    port = 5000

    mySocket = socket.socket()
    mySocket.connect((host, port))

    message = input(" -> ")

    while message != 'q':
        mySocket.send(message.encode())
        data = mySocket.recv(1024).decode()

        print('Received from server: ' + data)

        message = input(" -> ")

    mySocket.close()


if __name__ == '__main__':
    Main()
```

OUTPUT:



```
Connection from: ('127.0.0.1', 59593)
from connected  user: hello
sending: HELLO
from connected  user: world
sending: WORLD
```



```
 -> hello
Received from server: HELLO
 -> world
Received from server: WORLD
 ->
```

EXPERIMENT - 6

Write a program to create a server that listens to port 5006 using stream sockets. Write a simple client program to connect to the server. Run multiple clients that request the server for binary files. The server should service each client one after the other before terminating the connection.

Server code :

```
print("This is experiment 6th server code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
import socket
server=socket.socket()
server.bind(( "localhost" ,5006))
server.listen(3)
print( "Server Activated" )
while True:
    client, address = server.accept()
    print(str(address)+ "is connected.")
    client_request = client.recv(1024)
    print( 'Server received' ,client_request)
    filename=client_request
    f=open(filename,'rb')
    l=f.read(1024)
    client.send(l)
    f.close()
    print( "File Sent to "+ str(address)+ "server !")
server.close()
```

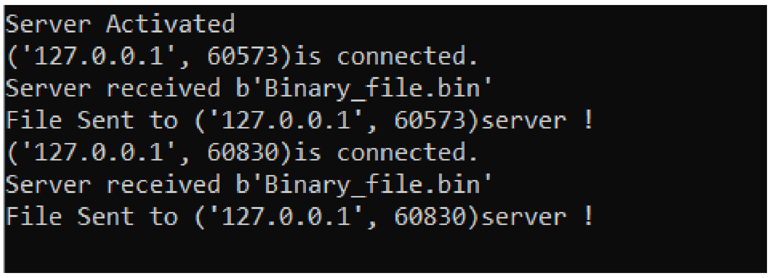
client1 code:

```
print("This is experiment 6th client1 code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
import socket
client=socket.socket()
source=input( "Enter desired file: " )
client.connect(( "localhost" ,5006))
client.send(source.encode())
n=input( "Enter new file name: " )
with open(n, "wb") as f:
   print( 'File opened' )
   while True:
     data=client.recv(1024)
     if not data:
      break
     print( 'Receiving data...' )
     print('Data in New file:')
     print(data)
     print( "file recieved by client01" )
     print( "Connection Lost" )
     f.close()
client.close()
```

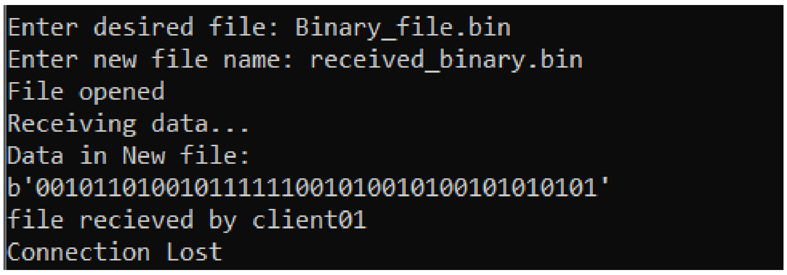
client2 code:

```
print("This is experiment 6th client2 code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
import socket
client=socket.socket()
source=input( "Enter desired file: " )
client.connect(( "localhost" ,5006))
client.send(source.encode())
n=input( "Enter new file name: " )
with open(n, "wb") as f:
   print( 'File opened' )
   while True:
     data=client.recv(1024)
     if not data:
      break
     print( 'Receiving data...' )
     print('Data in New file:')
     print(data)
     print( "file recieved by client01" )
     print( "Connection Lost" )
     f.close()
client.close()
```

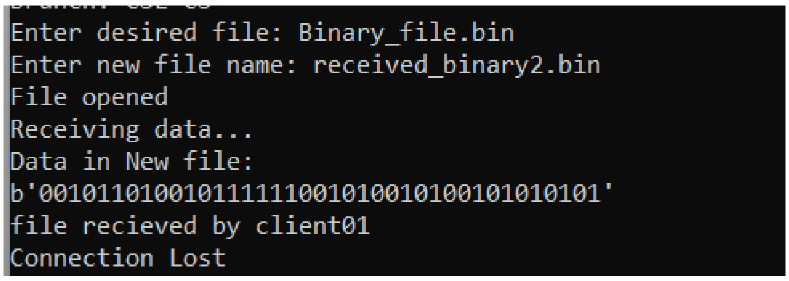
OUTPUT:



```
Server Activated
('127.0.0.1', 60573)is connected.
Server received b'Binary_file.bin'
File Sent to ('127.0.0.1', 60573)server !
('127.0.0.1', 60830)is connected.
Server received b'Binary_file.bin'
File Sent to ('127.0.0.1', 60830)server !
```



```
Enter desired file: Binary_file.bin
Enter new file name: received_binary.bin
File opened
Receiving data...
Data in New file:
b'001011010010111111001010010100101010101'
file recieved by client01
Connection Lost
```



```
Enter desired file: Binary_file.bin
Enter new file name: received_binary2.bin
File opened
Receiving data...
Data in New file:
b'001011010010111111001010010100101010101'
file recieved by client01
Connection Lost
```

EXPERIMENT - 7

Write a program to create a server that listens to port 5007 using stream sockets. Write a simple client program to connect to the server. Run multiple clients that request the server for text files. The server should service all clients concurrently.

Server code :

```
print("This is experiment 7th server code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
import socket
port=5007
s=socket.socket()
host=socket.gethostname()
s.bind((host,port))
s.listen(5)
print('Server is listening....')
while True:
    conn,addr=s.accept()
    print('Got connection from',addr)
    data=conn.recv(1024)
    print('Server received',data)
    filename=data
    f=open(filename,'rb')
    l=f.read(1024)
    while(l):
        conn.send(l)
        print('Sent',repr(l))
        l=f.read(1024)
        f.close
    print('Done sending' )
    conn.close()
```

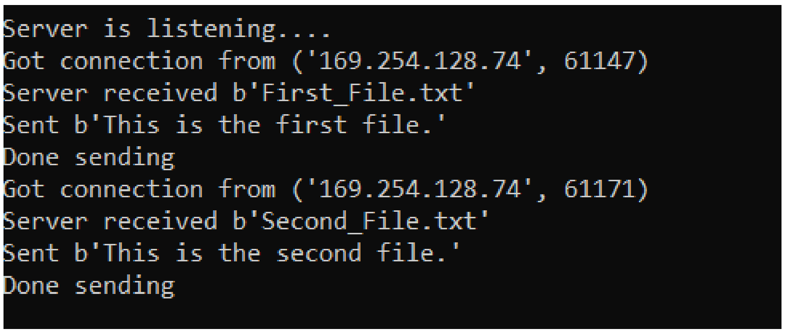
client1 code:

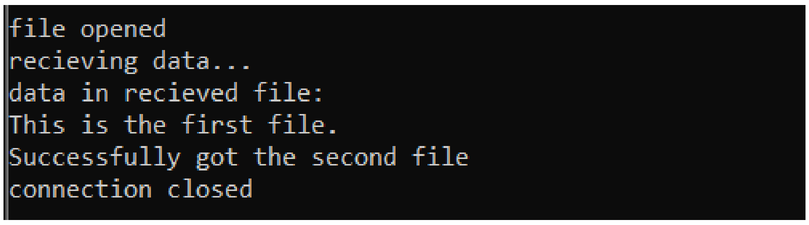
```
print("This is experiment 7th client1 code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
import socket
s=socket.socket()
host=socket.gethostname()
port=5007
s.connect((host,port))
s.send('First_File.txt'.encode())
with open('recieved_file.txt','wb')as f:
    print('file opened')
    while True:
        data=s.recv(1024).decode()
        if not data:
            break
        print('recieving data...')
        print('data in recieved file:')
        print(data)
        f.write(data.encode())
        f.close()
        print('Successfully got the second file')
s.close()
print('connection closed')
```

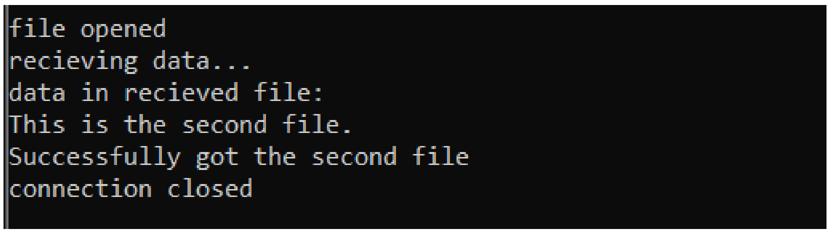
client2 code:

```
print("This is experiment 7th client2 code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
import socket
s=socket.socket()
host=socket.gethostname()
port=5007
s.connect((host,port))
s.send('Second_File.txt'.encode())
with open('recieved_file.txt','wb')as f:
    print('file opened')
    while True:
        data=s.recv(1024).decode()
        if not data:
            break
        print('recieving data...')
        print('data in recieved file:')
        print(data)
        f.write(data.encode())
        f.close()
        print('Successfully got the second file')
s.close()
print('connection closed')
```

OUTPUT:


```
Server is listening....
Got connection from ('169.254.128.74', 61147)
Server received b'First_File.txt'
Sent b'This is the first file.'
Done sending
Got connection from ('169.254.128.74', 61171)
Server received b'Second_File.txt'
Sent b'This is the second file.'
Done sending
```


```
file opened
recieving data...
data in recieved file:
This is the first file.
Successfully got the second file
connection closed
```


```
file opened
recieving data...
data in recieved file:
This is the second file.
Successfully got the second file
connection closed
```

EXPERIMENT - 8

Write a program to create a server that listens to port 5009 using datagram sockets. Write a simple client program that requests the server for a binary file. The server should service multiple clients concurrently and send the requested files in response.

<u>Server code :</u>

```
print("This is experiment 8 server code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
from socket import *
host= '127.0.0.1'
port=5009
server_soc= socket(AF_INET, SOCK_DGRAM)
server_soc.bind((host, port))
print( 'Server is ready to listen' )
n=int(input('Enter number of clients: '))
for i in range(n):
 file_name, client_adrs= server_soc.recvfrom(1024)
 print('Client',i+1,'- ', file_name.decode())
 try:
    f=open(file_name.decode(),'rb')
    l=f.read()
    server_soc.sendto(l, client_adrs)
    print('The file has been sent to client',i+1)
    f.close()
 except:
    msg='The file is not present'
    server_soc.sendto(msg.encode(), client_adrs)
```

client1 code:

```
print("This is experiment 8th
 client1 code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
from socket import *
host= '127.0.0.1'
port= 5009
client_soc= socket(AF_INET, SOCK_DGRAM)
file_name= input('Enter file name required: ')
client_soc.sendto(file_name.encode(),(host, port))
file, server_adrs= client_soc.recvfrom(1024)
```

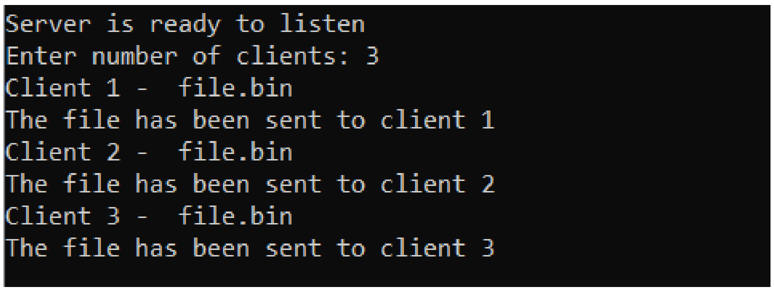
client2 code:

```
print("This is experiment 8th client2 code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
from socket import *
host= '127.0.0.1'
port= 5009
client_soc= socket(AF_INET, SOCK_DGRAM)
file_name= input('Enter file name required: ')
client_soc.sendto(file_name.encode(),(host, port))
file, server_adrs= client_soc.recvfrom(1024)
```

client3 code:

```
print("This is experiment 8th client3 code.")
print("Name: JASWANTH \nRoll No: 122010325012 \nBranch: CSE-CS")
from socket import *
host= '127.0.0.1'
port= 5009
client_soc= socket(AF_INET, SOCK_DGRAM)
file_name= input('Enter file name required: ')
client_soc.sendto(file_name.encode(),(host, port))
file, server_adrs= client_soc.recvfrom(1024)
```

OUTPUT:



```
Server is ready to listen
Enter number of clients: 3
Client 1 -  file.bin
The file has been sent to client 1
Client 2 -  file.bin
The file has been sent to client 2
Client 3 -  file.bin
The file has been sent to client 3
```

THANK YOU