PRINCIPLES OF NETWORKING – LAB REPORT 2

TASK 1:

EXPERIMENTAL SETUP:

The experimental setup consists of 8 Nodes connected through a Wireless Local Area Network in Ad-Hoc mode within a area defined by lower-left corner (x=-80 m, y=-80 m) and the upper-right corner (x=80 m, y=80 m). The parameters defined are as follows,
Channel: Default ns-3.
Physical Layer: IEEE802.11G and AARF Algorithm.
Link Layer: MAC without QOS.
Transport Layer: UDP

PROGRAM:

- Definition of the channel parameters, through which the nodes are going to transmit the data is done with the help of **YansWifiPhyHelper** and **YansWifiChannelHelper**.
- Standard MAC is defined without quality of service using **NqosWifiMacHelper.**
- All the nodes are set to Ad-Hoc mode using the module **AdhocWifiMac.**
- Since all nodes are wireless, we need to set up the mobility region of the nodes. This is done with the help of **Mobility Helper.**
- We then use the **InternetStackHelper** and **Ipv4AddressHelper** to install the network Stack and assign the IP address to the nodes.
- After assigning the IP addresses we define the attributes of the UDP echo application using the class **UdpEchoServerHelper** & **UdpEchoClientHelper** and define the client and server nodes.
- Since the topology consist of multiple networks, we use the **Ipv4GlobalRoutingHelper: PopulateRoutingTables()** to add a table for the routing purpose.

OBSERVATION:

- Details of the **CLIENT** (Node 7):
  - IP Address - 192.168.1.8
  - Client port - 49153
  - Transmits 2 UDP packets at time intervals of 1 and 2 seconds.

- Details of the **CLIENT** (Node 4):
  - IP Address - 192.168.1.5
  - Client port - 49153
  - Transmits 2 UDP packets at time intervals of 3 and 4 seconds.

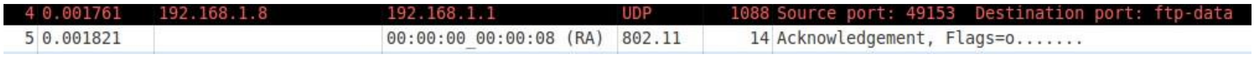
- Details of the **CLIENT** (Node 5):
  - IP Address - 192.168.1.6
  - Client port - 49153
  - Transmits 2 UDP packets at time intervals of 2 and 4 seconds.

- Details of the **SERVER** (Node 0):
  - IP Address - 192.168.1.1
  - Server port- 20

Frame Acknowledgement:

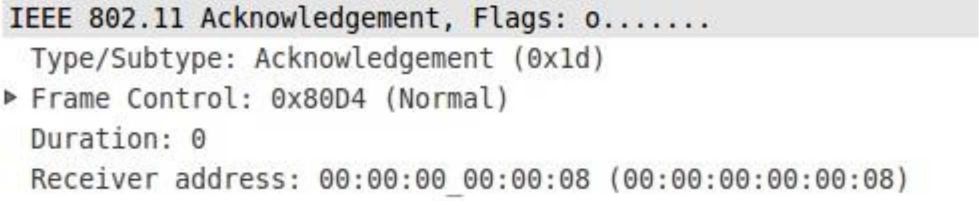
- WLAN IEEE-802.11 **uses CSMA Collision Avoidance with optional RTS/CTS.**
- CSMA/CA is **acknowledged and connectionless MAC protocol.**

- We know that in case of **CSMA/CA when Sender sends data packet to the Receiver node, it waits for Acknowledgement packet to be received from the Receiver** (Destination) to make sure that the Receiver as successfully received the data packet.
- If the **receiver does not send the acknowledgment packet and there is a time-out** at sender end, the **sender resends** the data packet assuming the first sent packet is lost/ didn't reach the receiver.
- For an Example, let us consider the scenario Node 7(client) sending packet to the Node 0(server) at 1 second.



```
4 0.001761   192.168.1.8   192.168.1.1              UDP     1088 Source port: 49153  Destination port: ftp-data
5 0.001821                 00:00:00_00:00:08 (RA)   802.11    14 Acknowledgement, Flags=o.......
```

- At time 1.001761 the source Node 192.168.1.8 sends a frame to the Destination Node 192.168.1.1.
- As soon as the Destination receives the frame it sends out the Acknowledgement frame to the Source. The Acknowledgment frame details are shown below,

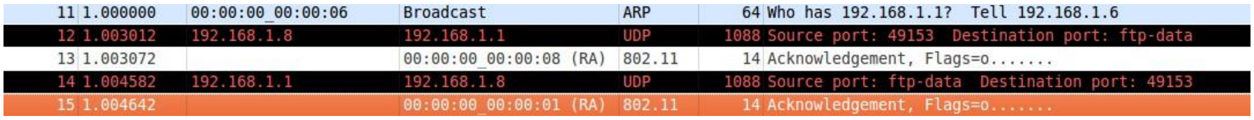


```
IEEE 802.11 Acknowledgement, Flags: o.......
  Type/Subtype: Acknowledgement (0x1d)
▶ Frame Control: 0x80D4 (Normal)
  Duration: 0
  Receiver address: 00:00:00_00:00:08 (00:00:00:00:00:08)
```

Collision:

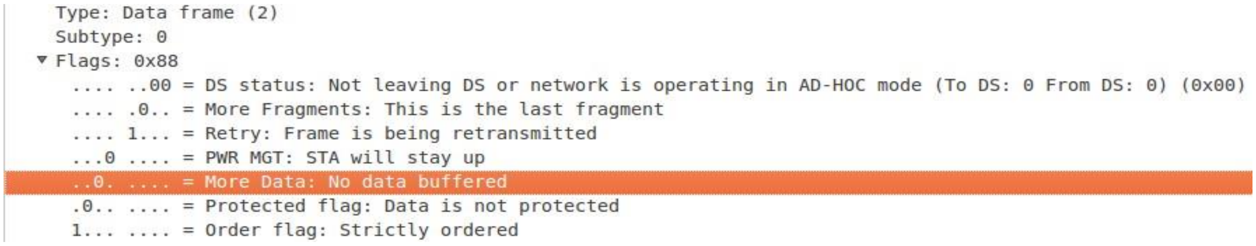
There is an instances where the collision occurs in the network. They are,

- At time **2 seconds**, **Node 7 and Node 5 starts transmission to the server at the same time. This leads to collision.** After going **for back-off Node 7 starts transmitting the packet successfully at 1.003012** and has successfully completed the transmission at 1.003072.



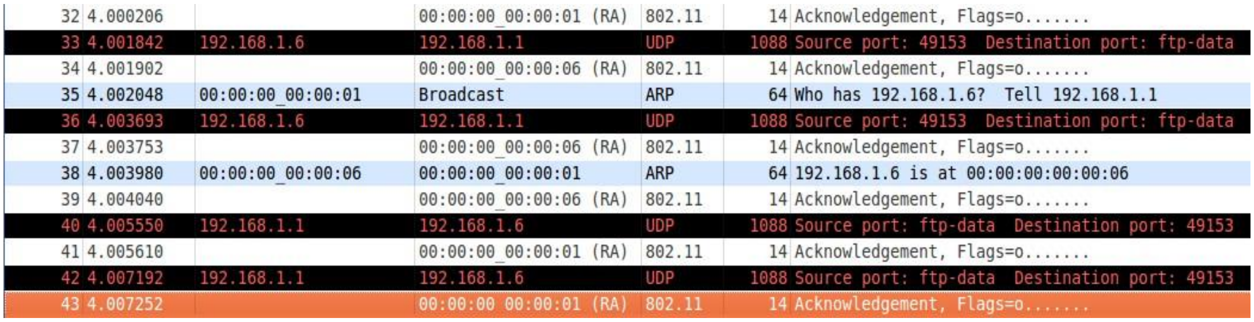
```
11 1.000000   00:00:00_00:00:06   Broadcast                ARP       64 Who has 192.168.1.1?  Tell 192.168.1.6
12 1.003012   192.168.1.8         192.168.1.1              UDP     1088 Source port: 49153  Destination port: ftp-data
13 1.003072                       00:00:00_00:00:08 (RA)   802.11    14 Acknowledgement, Flags=o.......
14 1.004582   192.168.1.1         192.168.1.8              UDP     1088 Source port: ftp-data  Destination port: 49153
15 1.004642                       00:00:00_00:00:01 (RA)   802.11    14 Acknowledgement, Flags=o.......
```

- In Wireshark we are also able to see that **the retry bit is set 1 when Node-7 retransmits the packet at time 1.003012.**



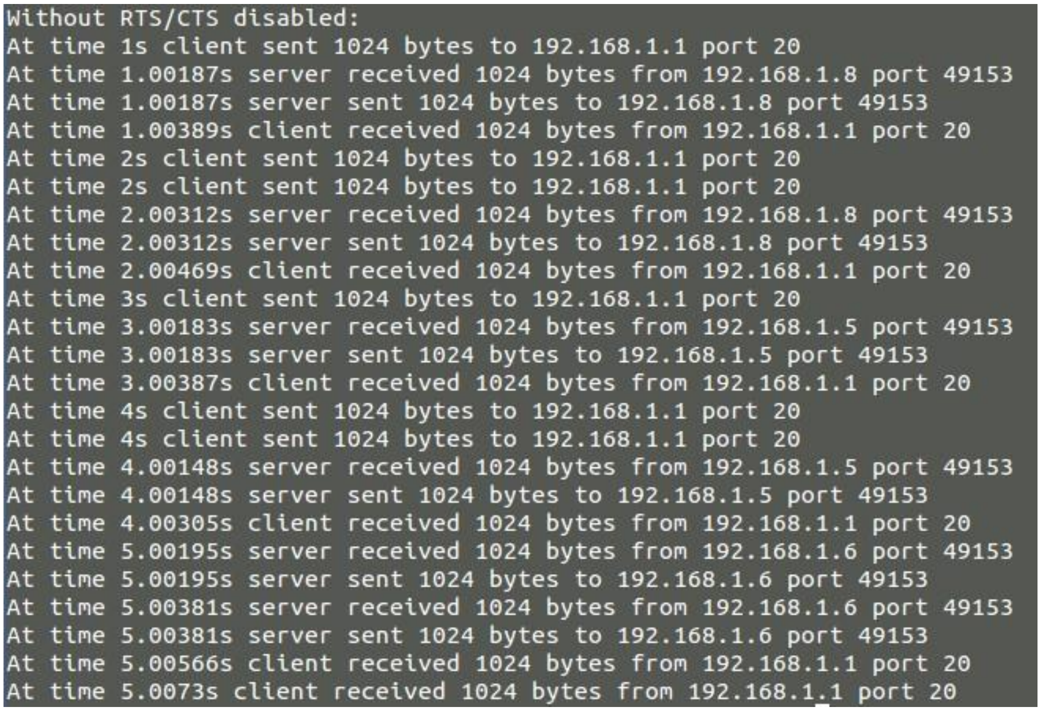
```
  Type: Data frame (2)
  Subtype: 0
▼ Flags: 0x88
    .... ..00 = DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0 From DS: 0) (0x00)
    .... .0.. = More Fragments: This is the last fragment
    .... 1... = Retry: Frame is being retransmitted
    ...0 .... = PWR MGT: STA will stay up
    ..0. .... = More Data: No data buffered
    .0.. .... = Protected flag: Data is not protected
    1... .... = Order flag: Strictly ordered
```

- But **Node 5 continues stay at back-off(multiple)** and gets its chance of transmitting the packet only at 5 seconds (after completing all the back-off policies).



32	4.000206		00:00:00_00:00:01 (RA)	802.11	14	Acknowledgement, Flags=o.......
33	4.001842	192.168.1.6	192.168.1.1	UDP	1088	Source port: 49153 Destination port: ftp-data
34	4.001902		00:00:00_00:00:06 (RA)	802.11	14	Acknowledgement, Flags=o.......
35	4.002048	00:00:00_00:00:01	Broadcast	ARP	64	Who has 192.168.1.6? Tell 192.168.1.1
36	4.003693	192.168.1.6	192.168.1.1	UDP	1088	Source port: 49153 Destination port: ftp-data
37	4.003753		00:00:00_00:00:06 (RA)	802.11	14	Acknowledgement, Flags=o.......
38	4.003980	00:00:00_00:00:06	00:00:00_00:00:01	ARP	64	192.168.1.6 is at 00:00:00:00:00:06
39	4.004040		00:00:00_00:00:06 (RA)	802.11	14	Acknowledgement, Flags=o.......
40	4.005550	192.168.1.1	192.168.1.6	UDP	1088	Source port: ftp-data Destination port: 49153
41	4.005610		00:00:00_00:00:01 (RA)	802.11	14	Acknowledgement, Flags=o.......
42	4.007192	192.168.1.1	192.168.1.6	UDP	1088	Source port: ftp-data Destination port: 49153
43	4.007252		00:00:00 00:00:01 (RA)	802.11	14	Acknowledgement, Flags=o.......

Output – RTS/CTS Disabled:



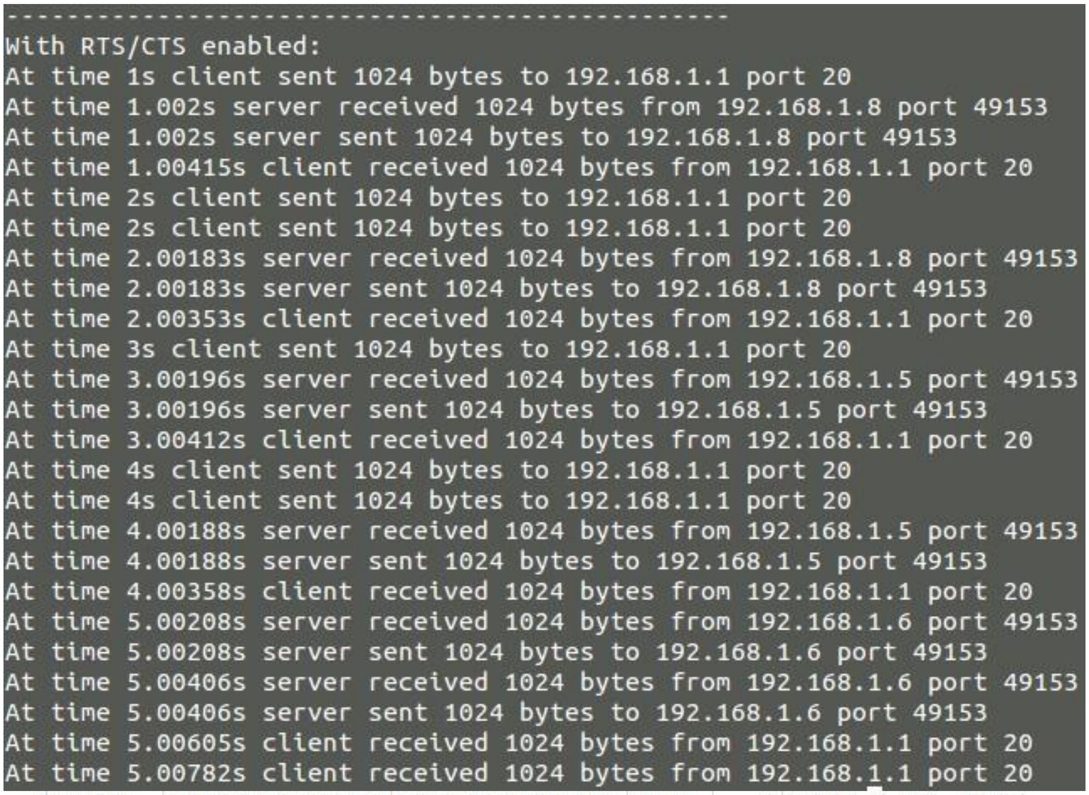
```
Without RTS/CTS disabled:
At time 1s client sent 1024 bytes to 192.168.1.1 port 20
At time 1.00187s server received 1024 bytes from 192.168.1.8 port 49153
At time 1.00187s server sent 1024 bytes to 192.168.1.8 port 49153
At time 1.00389s client received 1024 bytes from 192.168.1.1 port 20
At time 2s client sent 1024 bytes to 192.168.1.1 port 20
At time 2s client sent 1024 bytes to 192.168.1.1 port 20
At time 2.00312s server received 1024 bytes from 192.168.1.8 port 49153
At time 2.00312s server sent 1024 bytes to 192.168.1.8 port 49153
At time 2.00469s client received 1024 bytes from 192.168.1.1 port 20
At time 3s client sent 1024 bytes to 192.168.1.1 port 20
At time 3.00183s server received 1024 bytes from 192.168.1.5 port 49153
At time 3.00183s server sent 1024 bytes to 192.168.1.5 port 49153
At time 3.00387s client received 1024 bytes from 192.168.1.1 port 20
At time 4s client sent 1024 bytes to 192.168.1.1 port 20
At time 4s client sent 1024 bytes to 192.168.1.1 port 20
At time 4.00148s server received 1024 bytes from 192.168.1.5 port 49153
At time 4.00148s server sent 1024 bytes to 192.168.1.5 port 49153
At time 4.00305s client received 1024 bytes from 192.168.1.1 port 20
At time 5.00195s server received 1024 bytes from 192.168.1.6 port 49153
At time 5.00195s server sent 1024 bytes to 192.168.1.6 port 49153
At time 5.00381s server received 1024 bytes from 192.168.1.6 port 49153
At time 5.00381s server sent 1024 bytes to 192.168.1.6 port 49153
At time 5.00566s client received 1024 bytes from 192.168.1.1 port 20
At time 5.0073s client received 1024 bytes from 192.168.1.1 port 20
```

RTS/CTS Enabled:

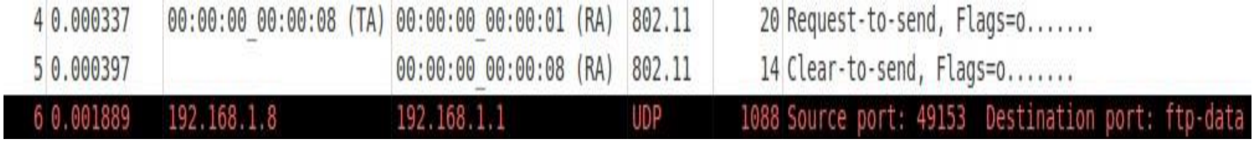
- In case of IEEE802.11 there is possibility of adding optional RTS/CTS. This is done with the help of below snippet of program,

UintegerValue ctsThr = (enableCtsRts ? UintegerValue (100) : UintegerValue (2200));
Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", ctsThr);

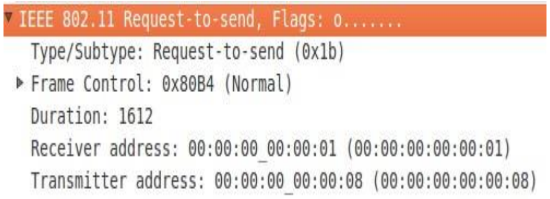
- If the value of **data packet is in the range of 100-2200 we enable the RTS/CTS** with the help of the **RTSCTSThreshold** module available in the **WifiRemoteRemoteStationManager module.**



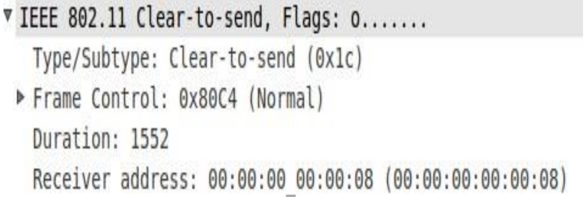
```
------------------------------------------------
With RTS/CTS enabled:
At time 1s client sent 1024 bytes to 192.168.1.1 port 20
At time 1.002s server received 1024 bytes from 192.168.1.8 port 49153
At time 1.002s server sent 1024 bytes to 192.168.1.8 port 49153
At time 1.00415s client received 1024 bytes from 192.168.1.1 port 20
At time 2s client sent 1024 bytes to 192.168.1.1 port 20
At time 2s client sent 1024 bytes to 192.168.1.1 port 20
At time 2.00183s server received 1024 bytes from 192.168.1.8 port 49153
At time 2.00183s server sent 1024 bytes to 192.168.1.8 port 49153
At time 2.00353s client received 1024 bytes from 192.168.1.1 port 20
At time 3s client sent 1024 bytes to 192.168.1.1 port 20
At time 3.00196s server received 1024 bytes from 192.168.1.5 port 49153
At time 3.00196s server sent 1024 bytes to 192.168.1.5 port 49153
At time 3.00412s client received 1024 bytes from 192.168.1.1 port 20
At time 4s client sent 1024 bytes to 192.168.1.1 port 20
At time 4s client sent 1024 bytes to 192.168.1.1 port 20
At time 4.00188s server received 1024 bytes from 192.168.1.5 port 49153
At time 4.00188s server sent 1024 bytes to 192.168.1.5 port 49153
At time 4.00358s client received 1024 bytes from 192.168.1.1 port 20
At time 5.00208s server received 1024 bytes from 192.168.1.6 port 49153
At time 5.00208s server sent 1024 bytes to 192.168.1.6 port 49153
At time 5.00406s server received 1024 bytes from 192.168.1.6 port 49153
At time 5.00406s server sent 1024 bytes to 192.168.1.6 port 49153
At time 5.00605s client received 1024 bytes from 192.168.1.1 port 20
At time 5.00782s client received 1024 bytes from 192.168.1.1 port 20
```



```
4 0.000337  00:00:00_00:00:08 (TA) 00:00:00_00:00:01 (RA) 802.11   20 Request-to-send, Flags=o.......
5 0.000397                         00:00:00_00:00:08 (RA) 802.11   14 Clear-to-send, Flags=o.......
6 0.001889  192.168.1.8            192.168.1.1            UDP    1088 Source port: 49153  Destination port: ftp-data
```



```
▾ IEEE 802.11 Request-to-send, Flags: o.......
    Type/Subtype: Request-to-send (0x1b)
  ▸ Frame Control: 0x80B4 (Normal)
    Duration: 1612
    Receiver address: 00:00:00_00:00:01 (00:00:00:00:00:01)
    Transmitter address: 00:00:00_00:00:08 (00:00:00:00:00:08)
```



```
▾ IEEE 802.11 Clear-to-send, Flags: o.......
    Type/Subtype: Clear-to-send (0x1c)
  ▸ Frame Control: 0x80C4 (Normal)
    Duration: 1552
    Receiver address: 00:00:00_00:00:08 (00:00:00:00:00:08)
```

The main **purpose of enforcing the RTS/CTS** connection is to,

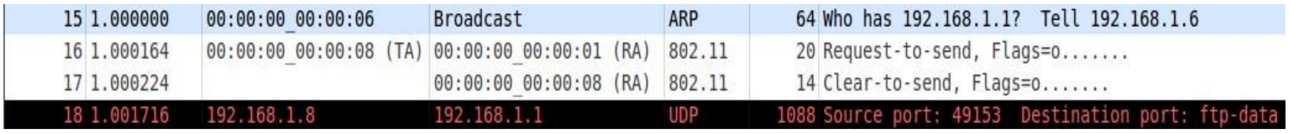
- **Hidden Terminal Problem**: In case of wireless communication most of the collision occurs due to the Hidden Terminal problem. Enforcing RTS/CTS helps us to overcome the hidden terminal problem.
- Enforcing RTS/CTS doesn't completely overcome the collision. **Collisions do occur**. But in this case **collision occurs only between the RTS packets which are of low data size**. This collision doesn't affect the overall efficiency of the network.

Collision after RTS/CTS:

- After enforcement of RTS/CTS there is **no collision between the data packet**. But there may be collision between the RTS packet if the Nodes starts to transmit at same time.
- **But in our case there is no collision of any packets**; i.e. irrespective of RTS/CTS.

Benefit of RTS/CTS:

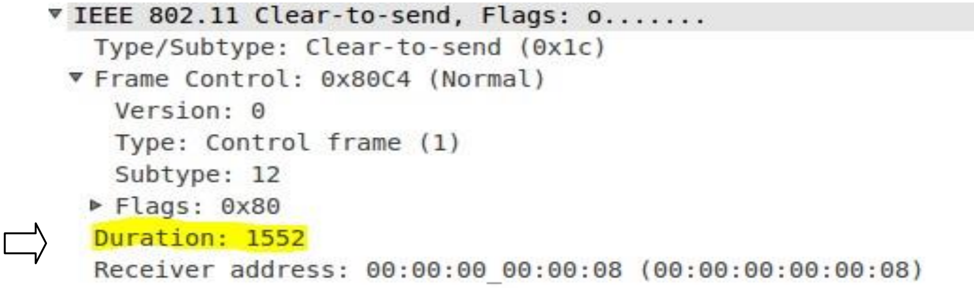
- In case of utilization RTS/CTS the possibility of having collision between the data packet is highly minimum.
- Even if the **collision occurs it is only between the RTS packet** which are of small size and doesn't affect the working of the network to a greater extent.
- Hence the probability of transmitting the data packet is high.
- In our case we can see that at time 2 seconds, **both Node 7 and 5 tries to transmit. But Node 7 sends the RTS packet before the Node 5(Since Node 5 takes some time to find the IP of the Server and in mean time Node 7 sends the RTS) and gets the access of the channel and successfully transmits the data.**



```
15 1.000000  00:00:00_00:00:06       Broadcast               ARP      64 Who has 192.168.1.1?  Tell 192.168.1.6
16 1.000164  00:00:00_00:00:08 (TA)  00:00:00_00:00:01 (RA)  802.11   20 Request-to-send, Flags=o.......
17 1.000224                          00:00:00_00:00:08 (RA)  802.11   14 Clear-to-send, Flags=o.......
18 1.001716  192.168.1.8             192.168.1.1             UDP    1088 Source port: 49153  Destination port: ftp-data
```

Network Allocation Vector:

- The NAV can be obtained from the **Frame Header of the MAC layer**. It is specified as Duration in the frame control.



```
▼ IEEE 802.11 Clear-to-send, Flags: o.......
    Type/Subtype: Clear-to-send (0x1c)
  ▼ Frame Control: 0x80C4 (Normal)
      Version: 0
      Type: Control frame (1)
      Subtype: 12
    ▶ Flags: 0x80
    Duration: 1552
    Receiver address: 00:00:00_00:00:08 (00:00:00:00:00:08)
```

- **In our case the NAV value is of 1552.**

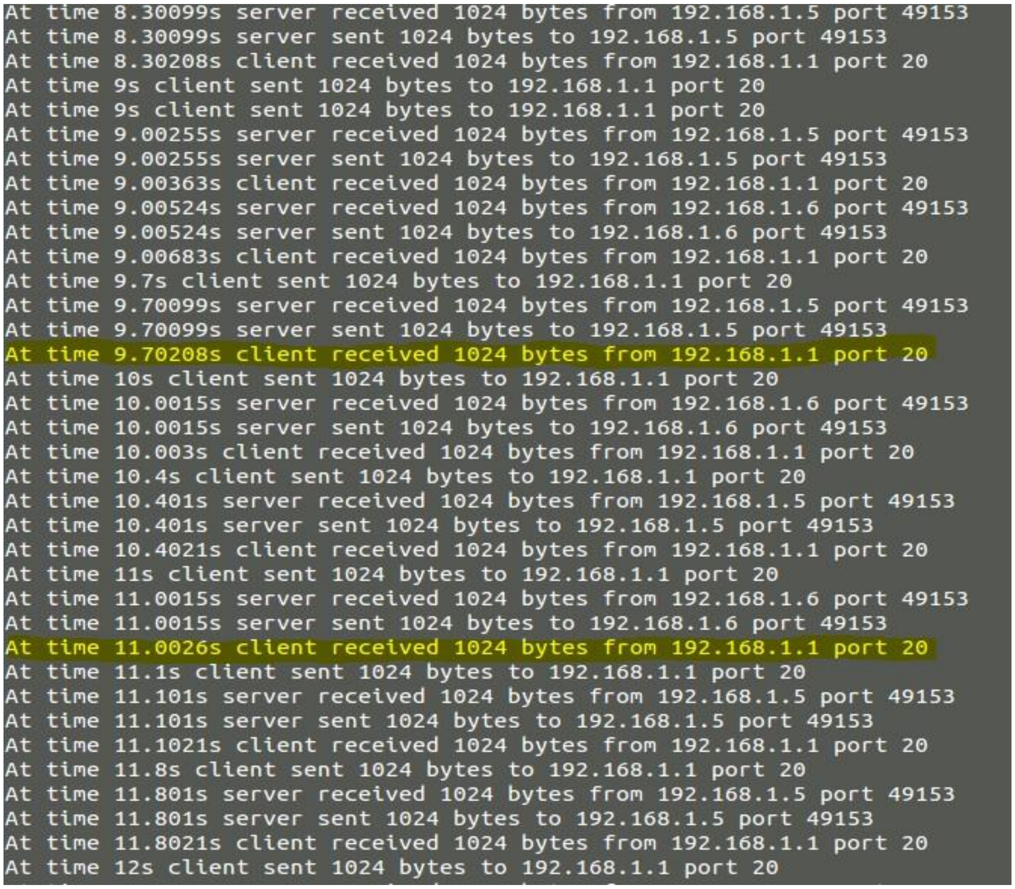
Enable NS_LOG_DEBUG and Changing the Application Layer to the new Requirement:

- The NS_LOG_DEBUG is enabled by sending **"true" value to the AarfWifiManager.**
  LogComponentEnable ("AarfWifiManager",LOG_DEBUG);

We are able to see that the transmission rate of the both nodes increases as it successfully completes transmission of data, up to the threshold.

- As we are using the AARF algorithm, the value of **transmission rate increases** when the number of **successful transmission threshold is reached.**
- In case of **Node 5**, the transmission rate which is **initially at 0.00305s increase to 0.0026 after the successful transmission of the 11th data packet. As there are more and more successful transmission the rate increases to 0.0018 and the last 25th packet has the transmission rate of 0.0016**.

- In case of **Node 4** the transmission rate of packet increases from **0.00305 to 0.0026 at 12th packet.** The node ends with transmission rate of **0.0018 for the 25th packet**.



```
At time 8.30099s server received 1024 bytes from 192.168.1.5 port 49153
At time 8.30099s server sent 1024 bytes to 192.168.1.5 port 49153
At time 8.30208s client received 1024 bytes from 192.168.1.1 port 20
At time 9s client sent 1024 bytes to 192.168.1.1 port 20
At time 9s client sent 1024 bytes to 192.168.1.1 port 20
At time 9.00255s server received 1024 bytes from 192.168.1.5 port 49153
At time 9.00255s server sent 1024 bytes to 192.168.1.5 port 49153
At time 9.00363s client received 1024 bytes from 192.168.1.1 port 20
At time 9.00524s server received 1024 bytes from 192.168.1.6 port 49153
At time 9.00524s server sent 1024 bytes to 192.168.1.6 port 49153
At time 9.00683s client received 1024 bytes from 192.168.1.1 port 20
At time 9.7s client sent 1024 bytes to 192.168.1.1 port 20
At time 9.70099s server received 1024 bytes from 192.168.1.5 port 49153
At time 9.70099s server sent 1024 bytes to 192.168.1.5 port 49153
At time 9.70208s client received 1024 bytes from 192.168.1.1 port 20
At time 10s client sent 1024 bytes to 192.168.1.1 port 20
At time 10.0015s server received 1024 bytes from 192.168.1.6 port 49153
At time 10.0015s server sent 1024 bytes to 192.168.1.6 port 49153
At time 10.003s client received 1024 bytes from 192.168.1.1 port 20
At time 10.4s client sent 1024 bytes to 192.168.1.1 port 20
At time 10.401s server received 1024 bytes from 192.168.1.5 port 49153
At time 10.401s server sent 1024 bytes to 192.168.1.5 port 49153
At time 10.4021s client received 1024 bytes from 192.168.1.1 port 20
At time 11s client sent 1024 bytes to 192.168.1.1 port 20
At time 11.0015s server received 1024 bytes from 192.168.1.6 port 49153
At time 11.0015s server sent 1024 bytes to 192.168.1.6 port 49153
At time 11.0026s client received 1024 bytes from 192.168.1.1 port 20
At time 11.1s client sent 1024 bytes to 192.168.1.1 port 20
At time 11.101s server received 1024 bytes from 192.168.1.5 port 49153
At time 11.101s server sent 1024 bytes to 192.168.1.5 port 49153
At time 11.1021s client received 1024 bytes from 192.168.1.1 port 20
At time 11.8s client sent 1024 bytes to 192.168.1.1 port 20
At time 11.801s server received 1024 bytes from 192.168.1.5 port 49153
At time 11.801s server sent 1024 bytes to 192.168.1.5 port 49153
At time 11.8021s client received 1024 bytes from 192.168.1.1 port 20
At time 12s client sent 1024 bytes to 192.168.1.1 port 20
```

The yellow highlight in the above the figure shows the increase in the transmission rate at Node 5 and Node 4 respectively.

We are able to see that the node-4 has its transmission rate increased first. Reason for this is below,

- AARF increases the transmission rate of the Node, when **the Nodes reach the Number of Successful transmission equal to that of the Threshold value set.**
- <u>**Node 4 increases the transmission rate first**</u> because **it transmits more number successful of packets in short duration, since interval between frame transmissions is 0.7s.**
- **At time 9.00363s the node 4 reaches the successful threshold set by the AARFWifiManager, hence it increases the transmission rate.**
- But the Node 5 increases its transmission time only at 10.003s.

TASK 2:

EXPERIMENTAL SETUP:

The setup consists of 8 Nodes operating under Infrastructure mode inclusive of 2 Access points, which communicate with the help of CSMA channel. The nodes defined in the Wireless Local Area Network (WLAN) are mobile and move in a rectangular two dimensional space. The nodes in the NETWORK1 as well as in the NETWORK2 section communicate through Access points using Wifi, whereas the Access points use CSMA channel in order to transfer data between nodes in two different networks.

- **192.168.1.0/24** -- The first set of nodes (Node 0,1,2) are assigned with this base address.
- **192.168.3.0/24** -- The second set of nodes (Node 3,4,5,6,7) are assigned with this base address.
- **192.168.2.0/24** -- The access points which are inter-connected using CSMA channel are assigned with this base address.

OBSERVATION:

- Details of the **CLIENT** (Node 1):
  - IP Address - 192.168.1.2
  - Client port - 49153
  - Transmits 2 UDP packets at time intervals of 2 and 4 seconds.

- Details of the **CLIENT** (Node 7):
  - IP Address - 192.168.3.5
  - Client port - 49153
  - Transmits 2 UDP packets at time intervals of 1 and 2 seconds.

- Details of the **CLIENT** (Node 5):
  - IP Address - 192.168.3.3
  - Client port - 49153
  - Transmits 2 UDP packets at time intervals of 3 and 4 seconds.

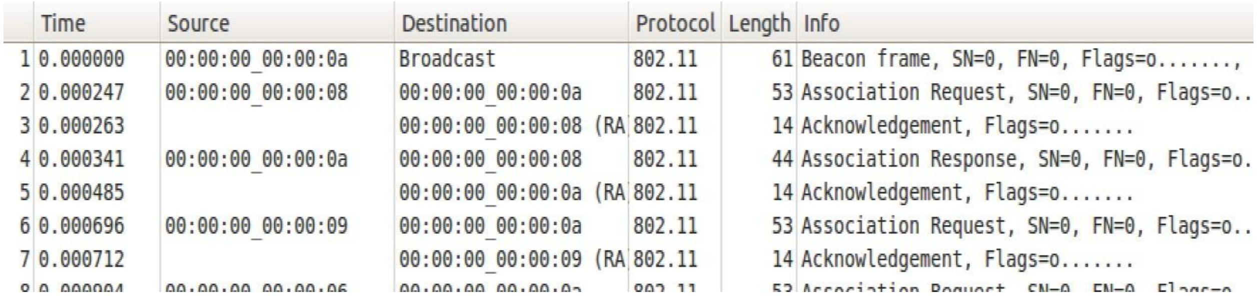
- Details of the **SERVER** (Node 0):
  - IP Address - 192.168.1.1
  - Server port- 20

PROGRAM:

- The channel parameters are defined in order to transmit data between nodes of different networks using **YansWifiPhyHelper** and **YansWifiChannelHelper.**
- The **SetRemoteStationManager** indicates the helper the type of rate control algorithm that is being used, which is AARF in our case.
- On the MAC layer, we have been asked to work with Non- QOS MACs, therefore we use **NqosWifiMacHelper** object to set the parameters.
- Once we are done with the MAC layer, we configure Wifi for all the devices or STA nodes using **NetDeviceContainer staDevices** and **apDevices** for access point nodes.
- The **Mobilityhelper** module is used in order to make the station nodes mobile within a specified region of a fixed dimension, which in our case is rectangular area defined by lower-left-corner (x=-80 m, y=-80 m) and the upper-right corner (x=80 m, y=80m).
- The **InternetStackHelper** and the **Ipv4AddressHelper** modules are used for setting up the stack layer assign the IP address to the nodes.
- The **UdpEchoServerHelper** and the **UdpEchoClientHelper** modules are used to define the client and server nodes in the networks.
- The **Ipv4GlobalRoutingHelper::PopulateRoutingTables()** is used to provide the routing table .

BEHAVIOUR OF ACCESS POINTS:

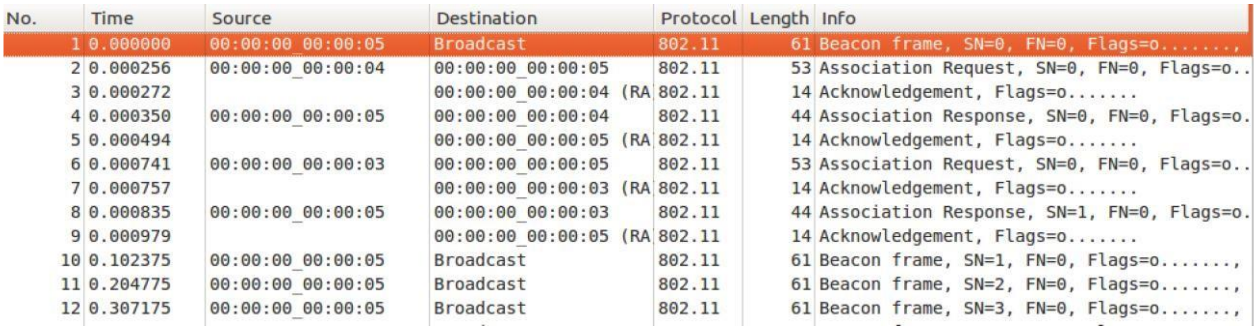
- Any **Data transmission between the nodes go through the Access Points**. It acts as control center of the network.
- By observing the packet tracer of the access points, we can infer that at the very first moment the network begins operating, the **Access Points broadcast beacon frames** so that the nodes within the specific network **are able to associate with the access points.**
- The Access points broadcasts the beacons when the channel remains idle at an **interval of every 0.102400 seconds.**
- The nodes become part of the network once the access point acknowledges the association request sent by the nodes to the access point.



	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:00:00_00:00:0a	Broadcast	802.11	61	Beacon frame, SN=0, FN=0, Flags=o.......,
2	0.000247	00:00:00_00:00:08	00:00:00_00:00:0a	802.11	53	Association Request, SN=0, FN=0, Flags=o..
3	0.000263		00:00:00_00:00:08 (RA	802.11	14	Acknowledgement, Flags=o.......
4	0.000341	00:00:00_00:00:0a	00:00:00_00:00:08	802.11	44	Association Response, SN=0, FN=0, Flags=o.
5	0.000485		00:00:00_00:00:0a (RA	802.11	14	Acknowledgement, Flags=o.......
6	0.000696	00:00:00_00:00:09	00:00:00_00:00:0a	802.11	53	Association Request, SN=0, FN=0, Flags=o..
7	0.000712		00:00:00_00:00:09 (RA	802.11	14	Acknowledgement, Flags=o.......
8	0.000904	00:00:00_00:00:06	00:00:00_00:00:0a	802.11	53	Association Request, SN=0, FN=0, Flags=o

fig. Packet tracer at Access point (NETWORK 2)

- In the above figure we can see that the access point with the physical address of 00:00:00_00:00:00:0a broadcasts a beacon frame the moment the network starts operating.
- At subsequent interval 0.000247s, 0.000696s, 0.000904, etc we see that other nodes that are part of the network send association requests to the access point. The above figure is with regard to the packet tracer at NETWORK2. The similar process happens at the Access point located at NETWORK1.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:00:00_00:00:05	Broadcast	802.11	61	Beacon frame, SN=0, FN=0, Flags=o.......,
2	0.000256	00:00:00_00:00:04	00:00:00_00:00:05	802.11	53	Association Request, SN=0, FN=0, Flags=o..
3	0.000272		00:00:00_00:00:04 (RA	802.11	14	Acknowledgement, Flags=o.......
4	0.000350	00:00:00_00:00:05	00:00:00_00:00:04	802.11	44	Association Response, SN=0, FN=0, Flags=o.
5	0.000494		00:00:00_00:00:05 (RA	802.11	14	Acknowledgement, Flags=o.......
6	0.000741	00:00:00_00:00:03	00:00:00_00:00:05	802.11	53	Association Request, SN=0, FN=0, Flags=o..
7	0.000757		00:00:00_00:00:03 (RA	802.11	14	Acknowledgement, Flags=o.......
8	0.000835	00:00:00_00:00:05	00:00:00_00:00:03	802.11	44	Association Response, SN=1, FN=0, Flags=o.
9	0.000979		00:00:00_00:00:05 (RA	802.11	14	Acknowledgement, Flags=o.......
10	0.102375	00:00:00_00:00:05	Broadcast	802.11	61	Beacon frame, SN=1, FN=0, Flags=o.......,
11	0.204775	00:00:00_00:00:05	Broadcast	802.11	61	Beacon frame, SN=2, FN=0, Flags=o.......,
12	0.307175	00:00:00_00:00:05	Broadcast	802.11	61	Beacon frame, SN=3, FN=0, Flags=o.......,

BEACON FRAME – PARAMETERS:

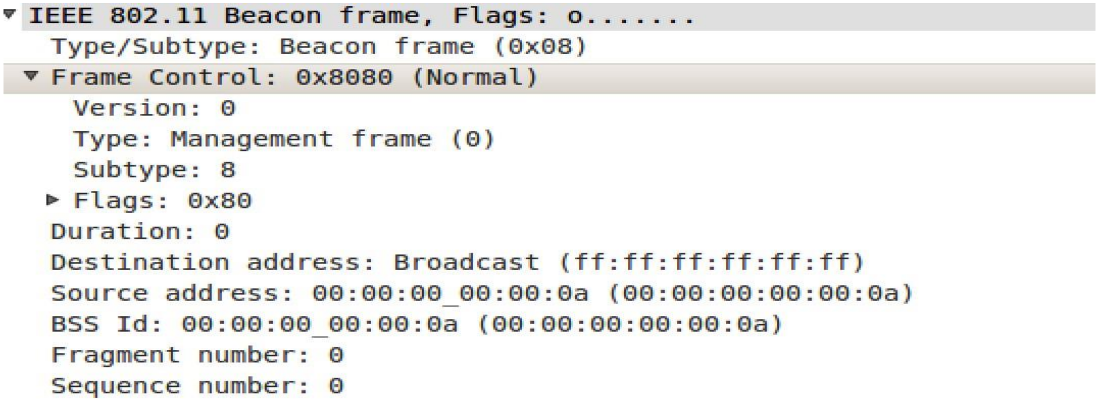
As mentioned before the beacon frames (around 60 Bytes) are broadcast by the Access Points at regular intervals of times, which is approximately equal to 0.1024s. The relevant contents of the beacon frame include Type/subtype, Frame control, Flags, Duration, Source and Destination MAC addresses, BSS ID, Followed by Fragment and Sequence Number.

- The frame control contains the version, type and subtype.

- **Flags:** The flag bits provide us with various insights with regard to the data that is being transmitted. Some of them are Order flag, Protected flag (If 0, it indicates that the data is not protected), DS status, PWR MGT, etc.
- **BSS ID**: Each Basic Service Set is having an Identification (ID). The BSS, operating in the infrastructure mode, the BSSID is nothing but the MAC address of the Access point.
- **Fragment and Sequence Number:** Fragment Number indicates the number of each frame sent of a fragmented frame and the sequence number indicates the sequence number of each frame.

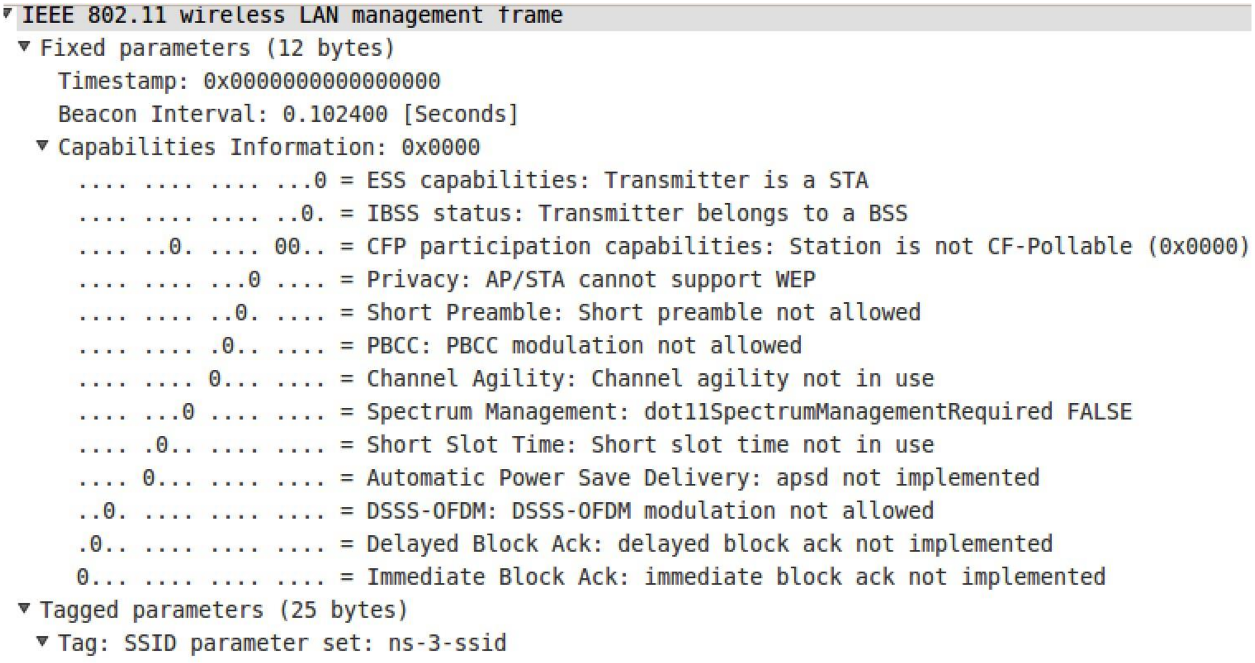
The frame body consists of the following parameters:

- **Timestamp (8 byte):** Timestamp is the time on AP, for which it has been active. When timestamp reaches its maximum value it will go back to 0. This field contain in Beacon Frame & Probe Response frame.
- **Beacon Interval (2 byte**) : Beacon Interval field time interval between target beacon transmission times. Default value is 102.4 milliseconds.
- **Capability Information (2 byte)** : This field contains number of subfields that are used to indicate requested or advertised optional capabilities.
- **SSID** : SSID is present in Beacons, probe requests, probe responses, association and re-association requests. Element ID is 0 for the SSID IE.
- **FH parameter set** : Used by legacy Frequency Hopping (FH) stations.
- **DS Parameter (2 byte)** : Present with beacon frame generated by stations using Clause 15, 18 or 19 PHY or if the beacon sent using one of the rates defined by one of the clause.
- **CF Parameter (8 byte)** : The CF parameter is used along with PCF.
- **IBSS parameter (4 byte):** It is present in beacon frames and it is generated by the stations nodes in IBSS .
- **TIM (Traffic Indication Map):** It is present in the beacon frames and is generated by the APs. It contains information for the nodes in low energy consuming mode. The AP makes use of Delivery Traffic Indication Map (DTIM) to inform the cell if it has frames buffered.



```
▼ IEEE 802.11 Beacon frame, Flags: o.......
   Type/Subtype: Beacon frame (0x08)
 ▼ Frame Control: 0x8080 (Normal)
     Version: 0
     Type: Management frame (0)
     Subtype: 8
   ▶ Flags: 0x80
   Duration: 0
   Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
   Source address: 00:00:00_00:00:0a (00:00:00:00:00:0a)
   BSS Id: 00:00:00_00:00:0a (00:00:00:00:00:0a)
   Fragment number: 0
   Sequence number: 0
```

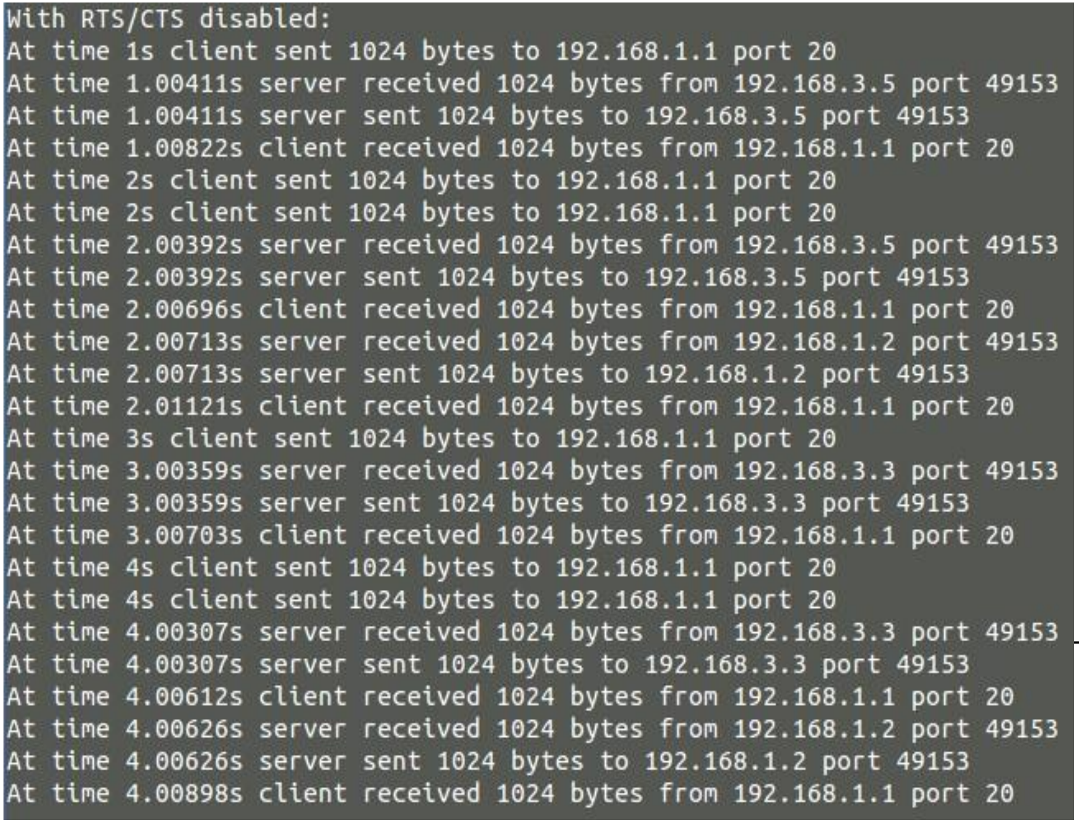
Fig: The below image shows the contents of the frame body.



```
▼ IEEE 802.11 wireless LAN management frame
 ▼ Fixed parameters (12 bytes)
     Timestamp: 0x0000000000000000
     Beacon Interval: 0.102400 [Seconds]
   ▼ Capabilities Information: 0x0000
       .... .... .... ...0 = ESS capabilities: Transmitter is a STA
       .... .... .... ..0. = IBSS status: Transmitter belongs to a BSS
       .... ..0. .... 00.. = CFP participation capabilities: Station is not CF-Pollable (0x0000)
       .... .... ...0 .... = Privacy: AP/STA cannot support WEP
       .... .... ..0. .... = Short Preamble: Short preamble not allowed
       .... .... .0.. .... = PBCC: PBCC modulation not allowed
       .... .... 0... .... = Channel Agility: Channel agility not in use
       .... ...0 .... .... = Spectrum Management: dot11SpectrumManagementRequired FALSE
       .... .0.. .... .... = Short Slot Time: Short slot time not in use
       .... 0... .... .... = Automatic Power Save Delivery: apsd not implemented
       ..0. .... .... .... = DSSS-OFDM: DSSS-OFDM modulation not allowed
       .0.. .... .... .... = Delayed Block Ack: delayed block ack not implemented
       0... .... .... .... = Immediate Block Ack: immediate block ack not implemented
 ▼ Tagged parameters (25 bytes)
   ▼ Tag: SSID parameter set: ns-3-ssid
```

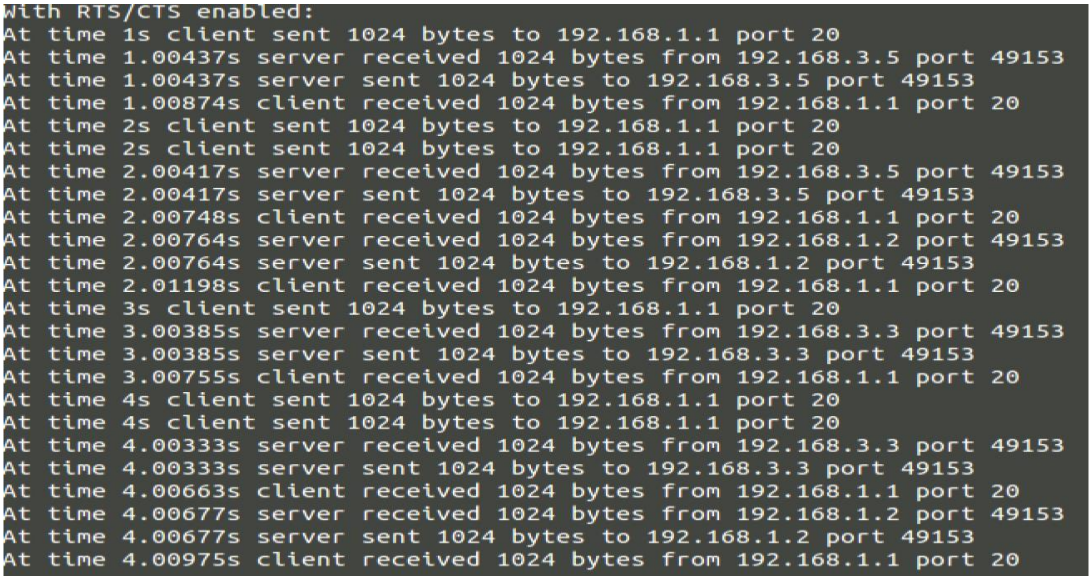
COLLISIONS WITH RTS/CTS DISABLED:

- With the feature of RTS/CTS being disabled we see there are no collisions taking place within the time frame of data transmissions between client nodes and the server node.
- Any Data transmission between the nodes go through the Access Points.



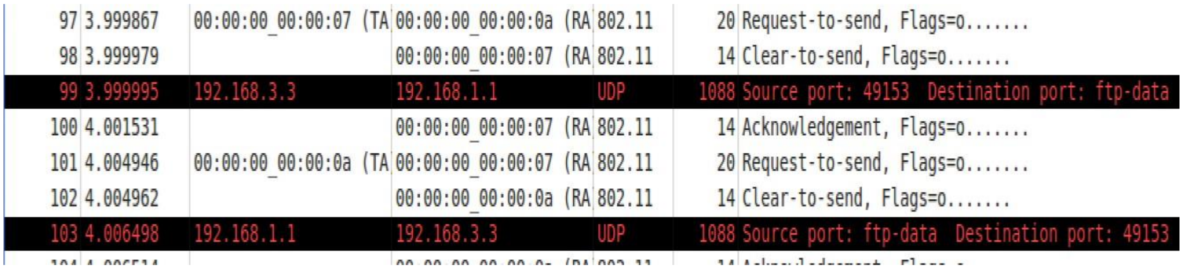
```
With RTS/CTS disabled:
At time 1s client sent 1024 bytes to 192.168.1.1 port 20
At time 1.00411s server received 1024 bytes from 192.168.3.5 port 49153
At time 1.00411s server sent 1024 bytes to 192.168.3.5 port 49153
At time 1.00822s client received 1024 bytes from 192.168.1.1 port 20
At time 2s client sent 1024 bytes to 192.168.1.1 port 20
At time 2s client sent 1024 bytes to 192.168.1.1 port 20
At time 2.00392s server received 1024 bytes from 192.168.3.5 port 49153
At time 2.00392s server sent 1024 bytes to 192.168.3.5 port 49153
At time 2.00696s client received 1024 bytes from 192.168.1.1 port 20
At time 2.00713s server received 1024 bytes from 192.168.1.2 port 49153
At time 2.00713s server sent 1024 bytes to 192.168.1.2 port 49153
At time 2.01121s client received 1024 bytes from 192.168.1.1 port 20
At time 3s client sent 1024 bytes to 192.168.1.1 port 20
At time 3.00359s server received 1024 bytes from 192.168.3.3 port 49153
At time 3.00359s server sent 1024 bytes to 192.168.3.3 port 49153
At time 3.00703s client received 1024 bytes from 192.168.1.1 port 20
At time 4s client sent 1024 bytes to 192.168.1.1 port 20
At time 4s client sent 1024 bytes to 192.168.1.1 port 20
At time 4.00307s server received 1024 bytes from 192.168.3.3 port 49153
At time 4.00307s server sent 1024 bytes to 192.168.3.3 port 49153
At time 4.00612s client received 1024 bytes from 192.168.1.1 port 20
At time 4.00626s server received 1024 bytes from 192.168.1.2 port 49153
At time 4.00626s server sent 1024 bytes to 192.168.1.2 port 49153
At time 4.00898s client received 1024 bytes from 192.168.1.1 port 20
```

- At time 4.00307s the server receives the frame from Node 5 of NETWORK2 and completes the UDP application process at 4.00898s when the client receives the packet from the server at 192.168.1.1.
- We use the option of RTS/CTS mode in order to overcome the **Hidden Terminal Problem.** Enforcing RTS/CTS as mentioned in earlier in adhoc problem, doesn't necessarily prevent collisions. But collisions in RTS/CTS mode has lesser effect on the overall efficiency of the network since the RTS frames are of smaller sizes.
- The following figure given below is the output for infrastructure mode of operation using **RTS/CTS option enabled**. Even with enabling the feature of RTS/CTS, while referring to the packet tracers of Node 5 (NETWORK2) and access points, we see there isn't any collisions taking place. Therefore, we can conclude that there are NO COLLISIONS taking place while enabling as well as disabling the RTS/CTS feature.



```
With RTS/CTS enabled:
At time 1s client sent 1024 bytes to 192.168.1.1 port 20
At time 1.00437s server received 1024 bytes from 192.168.3.5 port 49153
At time 1.00437s server sent 1024 bytes to 192.168.3.5 port 49153
At time 1.00874s client received 1024 bytes from 192.168.1.1 port 20
At time 2s client sent 1024 bytes to 192.168.1.1 port 20
At time 2s client sent 1024 bytes to 192.168.1.1 port 20
At time 2.00417s server received 1024 bytes from 192.168.3.5 port 49153
At time 2.00417s server sent 1024 bytes to 192.168.3.5 port 49153
At time 2.00748s client received 1024 bytes from 192.168.1.1 port 20
At time 2.00764s server received 1024 bytes from 192.168.1.2 port 49153
At time 2.00764s server sent 1024 bytes to 192.168.1.2 port 49153
At time 2.01198s client received 1024 bytes from 192.168.1.1 port 20
At time 3s client sent 1024 bytes to 192.168.1.1 port 20
At time 3.00385s server received 1024 bytes from 192.168.3.3 port 49153
At time 3.00385s server sent 1024 bytes to 192.168.3.3 port 49153
At time 3.00755s client received 1024 bytes from 192.168.1.1 port 20
At time 4s client sent 1024 bytes to 192.168.1.1 port 20
At time 4s client sent 1024 bytes to 192.168.1.1 port 20
At time 4.00333s server received 1024 bytes from 192.168.3.3 port 49153
At time 4.00333s server sent 1024 bytes to 192.168.3.3 port 49153
At time 4.00663s client received 1024 bytes from 192.168.1.1 port 20
At time 4.00677s server received 1024 bytes from 192.168.1.2 port 49153
At time 4.00677s server sent 1024 bytes to 192.168.1.2 port 49153
At time 4.00975s client received 1024 bytes from 192.168.1.1 port 20
```

- The figure below shows the packet tracer of Node 5 using Wireshark application.



97	3.999867	00:00:00_00:00:07 (TA	00:00:00_00:00:0a (RA	802.11	20	Request-to-send, Flags=o.......
98	3.999979		00:00:00_00:00:07 (RA	802.11	14	Clear-to-send, Flags=o.......
99	3.999995	192.168.3.3	192.168.1.1	UDP	1088	Source port: 49153 Destination port: ftp-data
100	4.001531		00:00:00_00:00:07 (RA	802.11	14	Acknowledgement, Flags=o.......
101	4.004946	00:00:00_00:00:0a (TA	00:00:00_00:00:07 (RA	802.11	20	Request-to-send, Flags=o.......
102	4.004962		00:00:00_00:00:0a (RA	802.11	14	Clear-to-send, Flags=o.......
103	4.006498	192.168.1.1	192.168.3.3	UDP	1088	Source port: ftp-data Destination port: 49153
104	4.006514		00:00:00_00:00:0a (RA	802.11	14	Acknowledgement, Flags=o