Second Homework Report

Antonio Ponti[*]
Edoardo Simonetti Spallotta[†]
Emmanuele Piola[‡]
Tania Silveri[§]

January, 4 2024

A full report on the second assigned Homework for the course "Fondamenti di Comunicazioni e Internet", Sapienza University of Rome.
Our working group is **Group 13**.

Contents

1 Description

The goal
of this project was to simulate, using the network simulator *ns-3*, the network seen in *Figure 1*. Tasked with this network, albeit not complex, we naturally opted for dividing it into smaller sub-networks, planning to link them at a later stage. We decided to use the following alphabetical identifiers for each sub-net:

- **Network A**: comprised of Nodes 10-16 and an Access Point (Node 9).
- **Network B**: comprised of Nodes 6, 7, 8 and a router (Node 5).
- **Network C**: comprised of Nodes 0, 1 and a router (Node 2).
- **Network D**: comprised of Node 3 and a router (Node 4).

We will refer to the links as **L1** (the network that links Network D and Network C), **L2** (Network A and Network D), and to **L3** (Network B and Network D). Our network had a total of four applications installed on different nodes, exchanging data via:

- **A TCP file delivery** of 1221MB,
  starting at 0.72s, from Node 7 (Net B) to Server 0 (Net C).

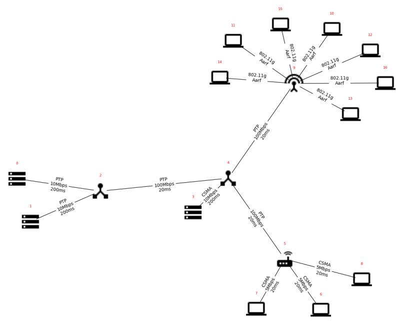


Figure 1: *Our network topology.*

[*]Student Number 2026296
[†]Student Number 2077561 - *group leader*
[‡]Student Number 2039978
[§]Student Number 2074824

- **A TCP file delivery** of 1264MB,
  starting at 3.93s, from Node 16 (Net A) to Server 1 (Net C).
- **A TCP file delivery** of 1754MB, starting at 3.87s, from Node 13 (Net A) to Server 0 (Net C).
- **A UDP Echo Application**, sending packets of 1964B, with a periodicity of 20ms, and the maximum number of packets being 250. The sender was Client 15 (Network A) and the receiver Server 3 (in Network D).

2 Network topology and delays

2.1 Known Topologies

We identified Network A as having a *star topology*, and the same goes for Network B, C. Note that a **single-layer tree topology** is closely related to a star topology, so it makes sense that Network C could be ambiguous as to whether it denotes a single layer tree topology rather than a star topology; while the distinction is more evident in Network A, since it has a larger number or spokes.
Finally, Network D, along with Networks L1-3, exhibits a simple *point-to-point (or linear) topology.*

2.2 Packets Path

We analyzed the traffic of each TCP application, highlighting the path taken by each packet, which was coherent with what we expected before the analysis. To exclude the UDP Echo Application, we used the **"tcp"** filter in Wireshark.

Table 1: TCP file delivery of 1221MB

NODE ID	**SENDER IP ADDRESS**	**RECEIVER IP ADDRESS**
Node 7 *(start)*	192.168.2.2	192.168.3.1
Node 5 (interface 2)	//	//
Node 4 (interface 2)	//	//
Node 2 (interface 0)	//	//
Node 0 *(end)*	//	192.168.3.1

Table 2: TCP file delivery of 1264MB

NODE ID	**SENDER IP ADDRESS**	**RECEIVER IP ADDRESS**
Node 16 *(start)*	192.168.1.7	192.168.3.5
Node 5 (interface 2)	//	//
Node 4 (interface 2)	//	//
Node 2 (interface 0)	//	//
Node 1 *(end)*	//	192.168.3.5

Table 3: TCP file delivery of 1754MB

NODE ID	**SENDER IP ADDRESS**	**RECEIVER IP ADDRESS**
Node 13 *(start)*	192.168.1.8	192.168.3.1
Node 5 (interface 2)	//	//
Node 4 (interface 2)	//	//
Node 2 (interface 0)	//	//
Node 0 *(end)*	//	192.168.3.1

2.3 Bottlenecks: Countermeasures and Solutions

From Fig. 1, it is apparent that the following are indeed *bottlenecks*:

- PTP link (Node 0 - Node 2)
- PTP link (Node 1 - Node 2)

- CSMA link (Node 6 - Node 5)
- CSMA link (Node 7 - Node 5)
- CSMA link (Node 8 - Node 5)
- CSMA link (Node 3 - Node 4)
- Each 802.11g link in Network A

There are multiple strategies to eliminate bottlenecks in a network, e.g. **increasing the link's bandwidth**, design priority packets and introduce network prioritization, **optimizing routing algorithms** to mitigate delays, improving network monitoring in order to act swiftly to intervene in case of malfunctioning, and to implement effective CA, CD protocols.

2.4 Average Throughput

Since we had to ignore UDP Echo Application traffic, we chose to analyze the throughput on Network L1 (using, again, the *"tcp"* filter). To isolate the packets sent up until t, we filtered packets based on their number, hence the filter used was *frame.number >= 0 && frame.number <= N*, where N represents the packet number at the requested time t.
At $t = 2.0s$, the calculated throughput is $0.100464 Mbps$, while at $t = 5.0s$, the calculated throughput amounted to $0.269821 Mbps$.
This is in line with the I/O Graph provided by Wireshark (Fig. 2), since the traffic at $t = 2.0s$ appears to be much less compared to the traffic at $t = 5.0s$. Note that we ignored the ACKs, since we wanted to focus on analyzing the data transmission to get a more accurate metric.

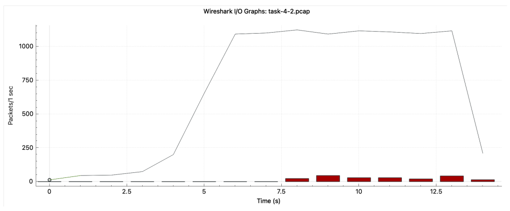


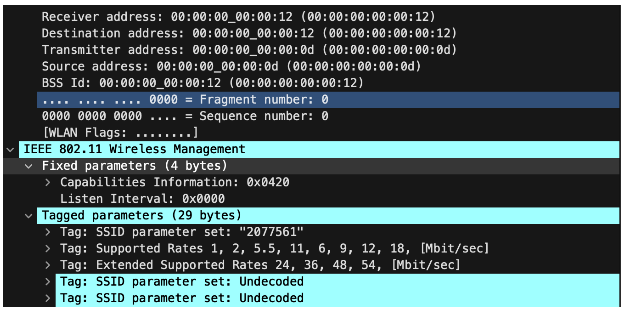
Figure 2: *I/O Graph for Node 4, interface 2*

3 Wireless Network, Infrastructure Mode

3.1 AP Behavior and Beacon Frame

We expected the Access Point (AP) to broadcast its Beacon frame as soon as the simulation started, followed by the Association Request (AReq) and its consequential ACK and Association Response (ARes), sent respectively by a Wi-Fi Station and the AP. We observed this exact behavior. Furthermore, the Beacon Frame provides important details:

- It tells each Station the AP's SSID, to which the AReq must be sent to.
- The DS flag indicates whether the AP is receiving or transmitting, respectively, it set to 01 or 10.
- The Payload is the Wireless Management, which includes important information such as whether Ad Hoc or Infrastructure Mode is used, and the available transmission rates.
- A timestamp to synchronize the Wi-Fi Stations.

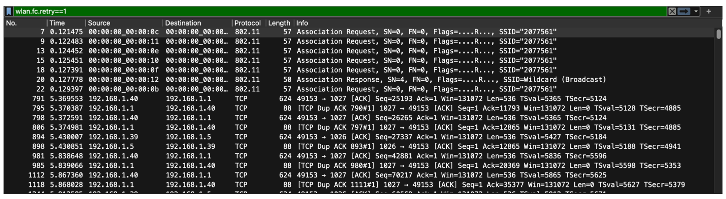


```
    Receiver address: 00:00:00_00:00:12 (00:00:00:00:00:12)
    Destination address: 00:00:00_00:00:12 (00:00:00:00:00:12)
    Transmitter address: 00:00:00_00:00:0d (00:00:00:00:00:0d)
    Source address: 00:00:00_00:00:0d (00:00:00:00:00:0d)
    BSS Id: 00:00:00_00:00:12 (00:00:00:00:00:12)
    .... .... .... 0000 = Fragment number: 0
    0000 0000 0000 .... = Sequence number: 0
    [WLAN Flags: ........]
v IEEE 802.11 Wireless Management
  v Fixed parameters (4 bytes)
    > Capabilities Information: 0x0420
      Listen Interval: 0x0000
  v Tagged parameters (29 bytes)
    > Tag: SSID parameter set: "2077561"
    > Tag: Supported Rates 1, 2, 5.5, 11, 6, 9, 12, 18, [Mbit/sec]
    > Tag: Extended Supported Rates 24, 36, 48, 54, [Mbit/sec]
    > Tag: SSID parameter set: Undecoded
    > Tag: SSID parameter set: Undecoded
```

Figure 3: *Screenshot of a Beacon frame Capture*

3.2 Collisions

When applying the *wlan.fc.retry==1* filter, to highlight every re-transmission at Node 9 (interface 1), many packets appear to be colliding (Fig. 4), e.g. the probable behavior of each Station after receiving the Beacon frame is that each one sends back the AReq frame at the same time, thereby causing collisions. Also, it's worth to note that the total number of packets that match the filter is 718. The cause for this high collision rate is that the RTS/CTS mechanism was not applied in this capture.



```
wlan.fc.retry==1
No.   Time     Source             Destination        Protocol Length Info
    7 0.121475 00:00:00_00:00:0c  00:00:00_00:00…    802.11     57 Association Request, SN=0, FN=0, Flags=....R..., SSID="2077561"
    9 0.122483 00:00:00_00:00:11  00:00:00_00:00…    802.11     57 Association Request, SN=0, FN=0, Flags=....R..., SSID="2077561"
   13 0.124452 00:00:00_00:00:0e  00:00:00_00:00…    802.11     57 Association Request, SN=0, FN=0, Flags=....R..., SSID="2077561"
   15 0.125451 00:00:00_00:00:10  00:00:00_00:00…    802.11     57 Association Request, SN=0, FN=0, Flags=....R..., SSID="2077561"
   18 0.127391 00:00:00_00:00:0f  00:00:00_00:00…    802.11     57 Association Request, SN=0, FN=0, Flags=....R..., SSID="2077561"
   20 0.127778 00:00:00_00:00:12  00:00:00_00:00…    802.11     50 Association Response, SN=4, FN=0, Flags=....R..., SSID=Wildcard (Broadcast)
   22 0.129397 00:00:00_00:00:0b  00:00:00_00:00…    802.11     57 Association Request, SN=0, FN=0, Flags=....R..., SSID="2077561"
  791 5.369553 192.168.1.40       192.168.1.1        TCP       624 49153 → 1027 [ACK] Seq=25193 Ack=1 Win=131072 Len=536 TSval=5365 TSecr=5124
  795 5.370387 192.168.1.1        192.168.1.40       TCP        88 [TCP Dup ACK 790#1] 1027 → 49153 [ACK] Seq=1 Ack=11793 Win=131072 Len=0 TSval=5128 TSecr=4885
  798 5.372591 192.168.1.40       192.168.1.1        TCP       624 49153 → 1027 [ACK] Seq=26265 Ack=1 Win=131072 Len=536 TSval=5365 TSecr=5124
  806 5.374981 192.168.1.1        192.168.1.40       TCP        88 [TCP Dup ACK 797#1] 1027 → 49153 [ACK] Seq=1 Ack=12865 Win=131072 Len=0 TSval=5131 TSecr=4885
  894 5.430007 192.168.1.39       192.168.1.5        TCP       624 49153 → 1026 [ACK] Seq=27337 Ack=1 Win=131072 Len=536 TSval=5427 TSecr=5184
  898 5.430851 192.168.1.5        192.168.1.39       TCP        88 [TCP Dup ACK 893#1] 1026 → 49153 [ACK] Seq=1 Ack=12865 Win=131072 Len=0 TSval=5188 TSecr=4941
  981 5.838648 192.168.1.40       192.168.1.1        TCP       624 49153 → 1027 [ACK] Seq=42881 Ack=1 Win=131072 Len=536 TSval=5836 TSecr=5596
  985 5.839066 192.168.1.1        192.168.1.40       TCP        88 [TCP Dup ACK 980#1] 1027 → 49153 [ACK] Seq=1 Ack=20369 Win=131072 Len=0 TSval=5598 TSecr=5353
 1112 5.867360 192.168.1.40       192.168.1.1        TCP       624 49153 → 1027 [ACK] Seq=70217 Ack=1 Win=131072 Len=536 TSval=5865 TSecr=5625
 1118 5.868028 192.168.1.1        192.168.1.40       TCP        88 [TCP Dup ACK 1111#1] 1027 → 49153 [ACK] Seq=1 Ack=35377 Win=131072 Len=0 TSval=5627 TSecr=5379
```

Figure 4: *Screenshot of the wlan.fc.retry==1 filter applied to the Node 9 - 1 capture,* **without RTS/CTS**

3.3 Overhead (No RTS/CTS)

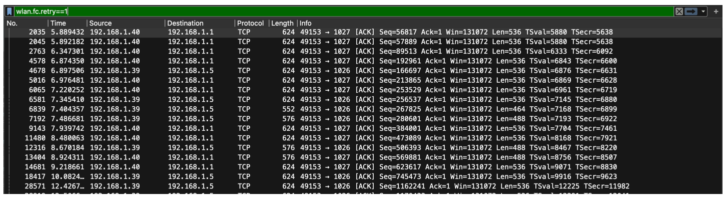
To calculate the total Overhead of our Echo Application, we used two filters (see Fig. (6)), namely:

- *wlan.addr == 00:00:00:00:00:0d || udp.*

The *00:00:00:00:00:0d* address refers to the sender Node in our application, and filters the Echo Reply. We then calculated the **overhead of each packet** and multiplied it by the number of packets that matched the filter, which resulted in a total of $18982B$.

3.4 RTS/CTS

As expected, the use of RTS/CTS eliminates the risk of collisions, which is due to the inherent mechanism of RTS/CTS. By enabling it, a Station that wants to send data first asks for permission to the AP, which replies with a broadcast message telling the requesting Station that it can start sending its frames. By broadcasting the CTS frame, each Station that has not sent an RTS knows that the AP is busy. Note that the total number of re-transmissions is now 47, and they could be caused by any other factor, e.g. congestion.



```
wlan.fc.retry==1
No.    Time      Source        Destination  Protocol Length Info
 2035  5.889432  192.168.1.40  192.168.1.1  TCP      624 49153 → 1027 [ACK] Seq=56817 Ack=1 Win=131072 Len=536 TSval=5880 TSecr=5638
 2045  5.892182  192.168.1.40  192.168.1.1  TCP      624 49153 → 1027 [ACK] Seq=57889 Ack=1 Win=131072 Len=536 TSval=5880 TSecr=5638
 2763  6.347301  192.168.1.40  192.168.1.1  TCP      624 49153 → 1027 [ACK] Seq=89513 Ack=1 Win=131072 Len=536 TSval=6333 TSecr=6092
 4578  6.874350  192.168.1.40  192.168.1.1  TCP      624 49153 → 1027 [ACK] Seq=192961 Ack=1 Win=131072 Len=536 TSval=6843 TSecr=6600
 4678  6.897506  192.168.1.39  192.168.1.5  TCP      624 49153 → 1026 [ACK] Seq=166697 Ack=1 Win=131072 Len=536 TSval=6876 TSecr=6631
 5016  6.976481  192.168.1.40  192.168.1.1  TCP      624 49153 → 1027 [ACK] Seq=213865 Ack=1 Win=131072 Len=536 TSval=6869 TSecr=6628
 6065  7.220252  192.168.1.40  192.168.1.1  TCP      624 49153 → 1027 [ACK] Seq=253529 Ack=1 Win=131072 Len=536 TSval=6961 TSecr=6719
 6581  7.345410  192.168.1.39  192.168.1.5  TCP      624 49153 → 1026 [ACK] Seq=256537 Ack=1 Win=131072 Len=536 TSval=7145 TSecr=6880
 6839  7.404357  192.168.1.39  192.168.1.5  TCP      552 49153 → 1026 [ACK] Seq=267825 Ack=1 Win=131072 Len=464 TSval=7168 TSecr=6899
 7192  7.486681  192.168.1.39  192.168.1.5  TCP      576 49153 → 1026 [ACK] Seq=280601 Ack=1 Win=131072 Len=488 TSval=7193 TSecr=6922
 9143  7.939742  192.168.1.40  192.168.1.1  TCP      624 49153 → 1027 [ACK] Seq=384001 Ack=1 Win=131072 Len=536 TSval=7704 TSecr=7461
11480  8.480063  192.168.1.40  192.168.1.1  TCP      624 49153 → 1027 [ACK] Seq=473089 Ack=1 Win=131072 Len=536 TSval=8168 TSecr=7921
12316  8.670184  192.168.1.39  192.168.1.5  TCP      576 49153 → 1026 [ACK] Seq=506393 Ack=1 Win=131072 Len=488 TSval=8467 TSecr=8220
13404  8.924311  192.168.1.40  192.168.1.1  TCP      576 49153 → 1027 [ACK] Seq=569881 Ack=1 Win=131072 Len=488 TSval=8756 TSecr=8507
14681  9.218661  192.168.1.40  192.168.1.1  TCP      624 49153 → 1027 [ACK] Seq=623617 Ack=1 Win=131072 Len=536 TSval=9071 TSecr=8830
18417  10.0824…  192.168.1.39  192.168.1.5  TCP      624 49153 → 1026 [ACK] Seq=745473 Ack=1 Win=131072 Len=536 TSval=9916 TSecr=9623
28571  12.4267…  192.168.1.39  192.168.1.5  TCP      624 49153 → 1026 [ACK] Seq=1162241 Ack=1 Win=131072 Len=536 TSval=12225 TSecr=11982
```

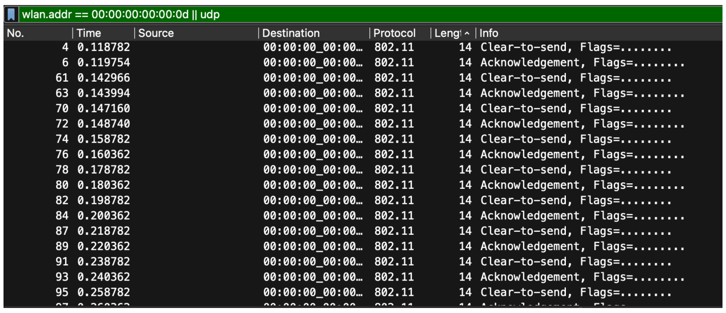
Figure 5: *Screenshot of the wlan.fc.retry==1 filter applied to the Node 9 - 1 capture,* **with RTS/CTS**

3.5 Overhead with RTS/CTS

Before tackling the calculation of the overhead, we should ask ourselves the question *"would it be any different?"* to which the **answer is obviously yes**. By enabling the RTS/CTS mechanism, we unavoidably generate more traffic, since the nodes now have to exchange additional frames. To identify which frames to consider, we used the *wlan.addr == 00:00:00:00:00:0d || udp* filter to isolate the packets we were interested in analyzing, and added each individual overhead:

$$(14 * 490) + (20 * 245) + (64 * 243) = 27312B \qquad (1)$$

In conclusion, the total overhead with RTS/CTS is equal to $27312B > 18982B$.



```
wlan.addr == 00:00:00:00:00:0d || udp
No.  Time      Source  Destination      Protocol Length Info
  4  0.118782          00:00:00_00:00…  802.11   14 Clear-to-send, Flags=........
  6  0.119754          00:00:00_00:00…  802.11   14 Acknowledgement, Flags=........
 61  0.142966          00:00:00_00:00…  802.11   14 Clear-to-send, Flags=........
 63  0.143994          00:00:00_00:00…  802.11   14 Acknowledgement, Flags=........
 70  0.147160          00:00:00_00:00…  802.11   14 Clear-to-send, Flags=........
 72  0.148740          00:00:00_00:00…  802.11   14 Acknowledgement, Flags=........
 74  0.158782          00:00:00_00:00…  802.11   14 Clear-to-send, Flags=........
 76  0.160362          00:00:00_00:00…  802.11   14 Acknowledgement, Flags=........
 78  0.178782          00:00:00_00:00…  802.11   14 Clear-to-send, Flags=........
 80  0.180362          00:00:00_00:00…  802.11   14 Acknowledgement, Flags=........
 82  0.198782          00:00:00_00:00…  802.11   14 Clear-to-send, Flags=........
 84  0.200362          00:00:00_00:00…  802.11   14 Acknowledgement, Flags=........
 87  0.218782          00:00:00_00:00…  802.11   14 Clear-to-send, Flags=........
 89  0.220362          00:00:00_00:00…  802.11   14 Acknowledgement, Flags=........
 91  0.238782          00:00:00_00:00…  802.11   14 Clear-to-send, Flags=........
 93  0.240362          00:00:00_00:00…  802.11   14 Acknowledgement, Flags=........
 95  0.258782          00:00:00_00:00…  802.11   14 Clear-to-send, Flags=........
```

Figure 6: *Screenshot of the wlan.addr == 00:00:00:00:00:0d || udp filter applied to the Node 15 capture,* **with RTS/CTS**