UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

NETWORK SIMULATION

Lecturer: Mrs. Nguyen Minh Huong

An Evaluation of CSMA/CA Protocol Without RTS/CTS in Ad-hoc WLAN Network

Members:

Hà Trung Tín	BI11-261
Lê Phương Thu	BI11-258
Hà Thu An	BI11-001
Nguyễn Đức Anh Tuấn	BA10-066
Nguyễn Thị Khánh Huyền	BA10-025
Phạm Huy Thiệp	BI11-255

March, 2023

Problems

- Consider the CSMA/CA protocol in Wifi networks that work in ad-hoc mode.
- Evaluate the performance of the protocol without RTS/CTS scheme when the number of nodes within a communication range increases from 2 to 30.

Table of Contents

I. INTRODUCTION

In 1974, the term “Internet” first appeared, but at that time people still used the word ARPANET. It was not until the 1995s that the Internet gradually became more familiar when the ARPANET network was no longer effective. The first Internet inventor was Paul Baran with two other colleagues, Donald Davies and Leonard Kleinrock. He formed the first network of links, which are Stanford Research Institute, the University of California, Los Angeles, the University of Utah, and the University of California, Santa Barbara of the US, allowing the exchange of data belonging to these regions.

After a long time of research and development, Vietnam’s Internet history recorded November 19, 1997 as the day when the S-shaped country connected to the world’s information highway. During the past 20 years. The Internet has had a direct impact, changing many conceptions and lifestyles of Vietnamese people.

In recent years, wireless network technology has developed rapidly and covered almost all countries in the world, however, the increase in the number of such network devices has raised many problems related to the optimization of the transmission line and causing collisions.

In fact, CSMA/CA (Carrier sense multiple access/collision avoidance) has been applied in basic 802.11 networks. Thanks to RTS/CTS (request to send/ clear to send), some problems can be resolved such as devices in a wireless network may appear to be concealed from one another, making it more difficult for individual devices to determine when to broadcast.

In this research, we will use NS-3.34 to simulate an ad-hoc wireless network utilizing the CSMA/CA protocol without RTS/CTS to determine the efficiency of RTS/CTS.)

Wireless networks have become ubiquitous in our daily lives and are an essential means of communication. One of the most commonly used wireless networking protocols is CSMA/CA, which is used in Wi-fi networks operating in Ad-hoc Mode.

II. RESEARCH AND REVIEW

CSMA/CA

- It is a transmission access mechanism in a wireless network environment, what about CSMA/CD is the transmission access mechanism in wired LAN?

So, we use CSMA/CA with many specifications similar to CSMA/CD.

RTS/CTS

The RTS/CTS (Request to Send / Clear to Send) mechanism is a flow control method used to avoid data collision in wireless networks. RTS/CTS is particularly useful in environments with multiple devices competing for the wireless medium. It can be simulated in network simulations by adjusting parameters to evaluate its effectiveness in different scenarios.

Ad-hoc Mode

- It is a wireless networking mode where devices communicate directly with each other without the use of an access point or router.
- It allows wireless devices to form a network on the fly without the need for an existing network infrastructure.

In this project, we will analyze and evaluate the performance of the CSMA/CA protocol without the RTS/CTS scheme in Wi-fi networks, operating in Ad-hoc Mode as the number of nodes within the communication range increase from 2-30.

III. SET UP APPLICATION

- *Design*

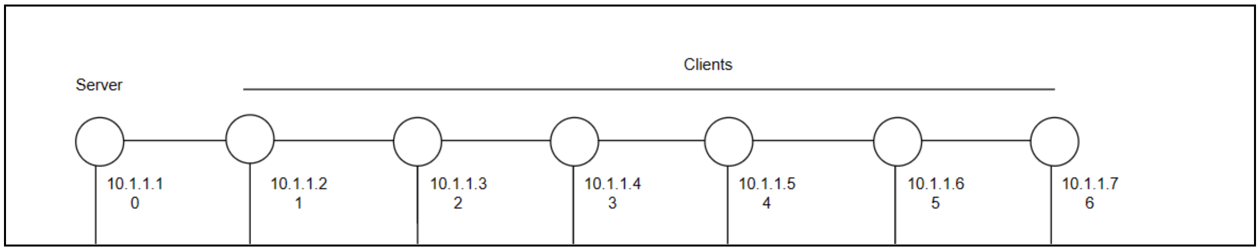
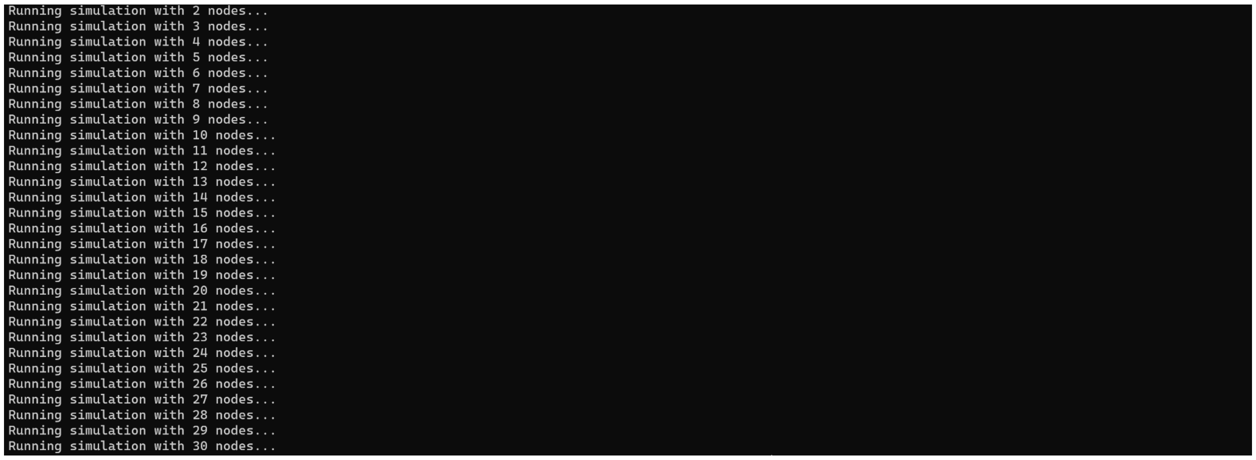


Figure 1. An example of a design with 7 nodes

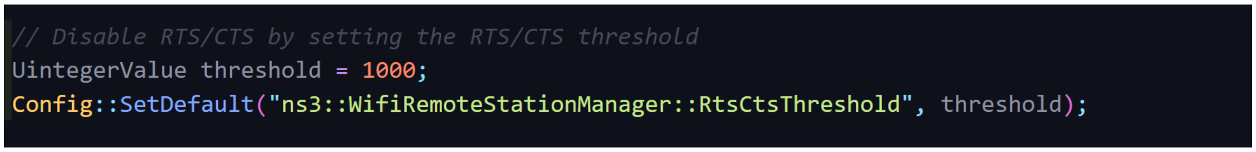
- 1 node is a server (default is the 1st node).
- The rest are clients.
- All clients will simultaneously send packets to the server.

- *Running simulation with 2-30 nodes*



```
Running simulation with 2 nodes...
Running simulation with 3 nodes...
Running simulation with 4 nodes...
Running simulation with 5 nodes...
Running simulation with 6 nodes...
Running simulation with 7 nodes...
Running simulation with 8 nodes...
Running simulation with 9 nodes...
Running simulation with 10 nodes...
Running simulation with 11 nodes...
Running simulation with 12 nodes...
Running simulation with 13 nodes...
Running simulation with 14 nodes...
Running simulation with 15 nodes...
Running simulation with 16 nodes...
Running simulation with 17 nodes...
Running simulation with 18 nodes...
Running simulation with 19 nodes...
Running simulation with 20 nodes...
Running simulation with 21 nodes...
Running simulation with 22 nodes...
Running simulation with 23 nodes...
Running simulation with 24 nodes...
Running simulation with 25 nodes...
Running simulation with 26 nodes...
Running simulation with 27 nodes...
Running simulation with 28 nodes...
Running simulation with 29 nodes...
Running simulation with 30 nodes...
```

Figure 2. Running the simulation with 2-30 nodes

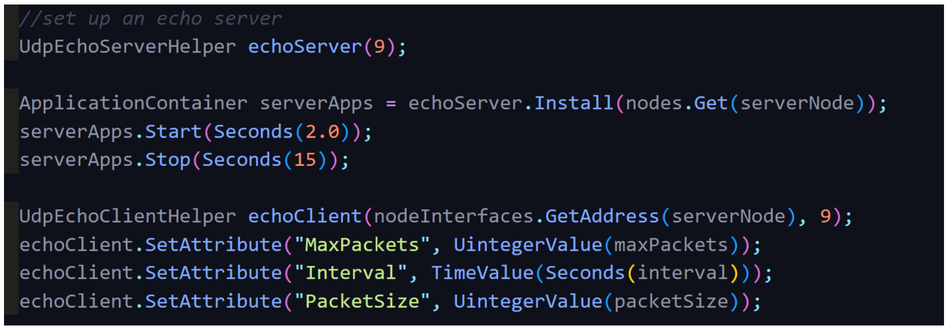


```cpp
// Disable RTS/CTS by setting the RTS/CTS threshold
UintegerValue threshold = 1000;
Config::SetDefault("ns3::WifiRemoteStationManager::RtsCtsThreshold", threshold);
```

Figure 3. Disable RTS/CTS by setting the threshold

➔ *Configuration of* `UdpEchoServer`:

- Firstly, we create a `UdpEchoServerHelper` and provide the server port number.
- Secondly, instantiate the server on the chosen server node.

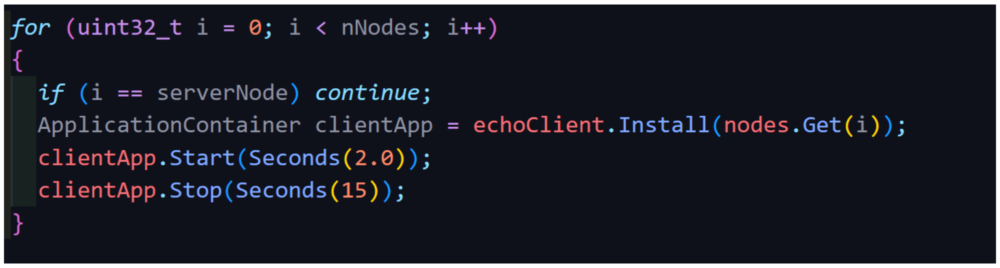


```cpp
//set up an echo server
UdpEchoServerHelper echoServer(9);

ApplicationContainer serverApps = echoServer.Install(nodes.Get(serverNode));
serverApps.Start(Seconds(2.0));
serverApps.Stop(Seconds(15));

UdpEchoClientHelper echoClient(nodeInterfaces.GetAddress(serverNode), 9);
echoClient.SetAttribute("MaxPackets", UintegerValue(maxPackets));
echoClient.SetAttribute("Interval", TimeValue(Seconds(interval)));
echoClient.SetAttribute("PacketSize", UintegerValue(packetSize));
```

Figure 4. Set up an echo server

➔ *Configuration of* `UdpEchoClient`:

- Create a `UdpEchoClientHelper` and provide the remote address and port.
- After, we install the client on every other node except the serverNode.



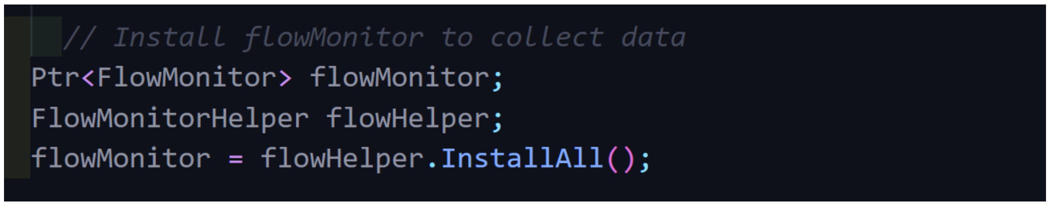
```
for (uint32_t i = 0; i < nNodes; i++)
{
  if (i == serverNode) continue;
  ApplicationContainer clientApp = echoClient.Install(nodes.Get(i));
  clientApp.Start(Seconds(2.0));
  clientApp.Stop(Seconds(15));
}
```

Figure 5. Setup the Running time

IV. *COLLECT AND ANALYZE DATA*

1. *Collect Data*

- We use a flow monitor as the main method to collect data.

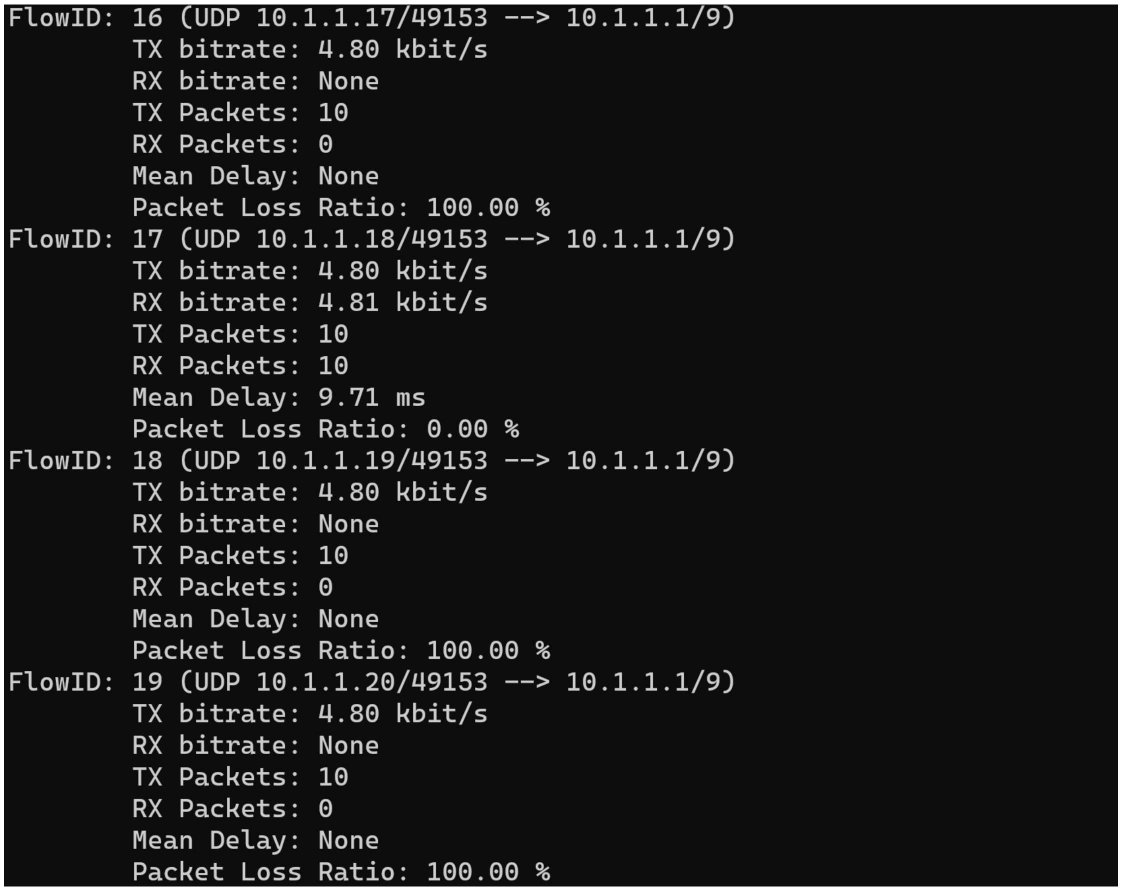


```
  // Install flowMonitor to collect data
Ptr<FlowMonitor> flowMonitor;
FlowMonitorHelper flowHelper;
flowMonitor = flowHelper.InstallAll();
```

Figure 6. Install flowMonitor to collect data

2. *Analyze Data*

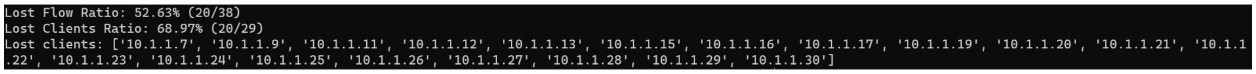
- *Data from Flow Monitor*: Data is stored in flows that contain all data about packets sent by a particular host to another.
- *Data from the flows contains a lot of information,* including:
  - The time the first and the last packet is transmitted and received.
  - The total delay.
  - The total bytes and packets transmitted and received.
  - Number of lost packets.



```
FlowID: 16 (UDP 10.1.1.17/49153 --> 10.1.1.1/9)
        TX bitrate: 4.80 kbit/s
        RX bitrate: None
        TX Packets: 10
        RX Packets: 0
        Mean Delay: None
        Packet Loss Ratio: 100.00 %
FlowID: 17 (UDP 10.1.1.18/49153 --> 10.1.1.1/9)
        TX bitrate: 4.80 kbit/s
        RX bitrate: 4.81 kbit/s
        TX Packets: 10
        RX Packets: 10
        Mean Delay: 9.71 ms
        Packet Loss Ratio: 0.00 %
FlowID: 18 (UDP 10.1.1.19/49153 --> 10.1.1.1/9)
        TX bitrate: 4.80 kbit/s
        RX bitrate: None
        TX Packets: 10
        RX Packets: 0
        Mean Delay: None
        Packet Loss Ratio: 100.00 %
FlowID: 19 (UDP 10.1.1.20/49153 --> 10.1.1.1/9)
        TX bitrate: 4.80 kbit/s
        RX bitrate: None
        TX Packets: 10
        RX Packets: 0
        Mean Delay: None
        Packet Loss Ratio: 100.00 %
```

Figure 7. An example of data with 4 nodes

=> ***All packets sent by some clients are completely lost.***



```
Lost Flow Ratio: 52.63% (20/38)
Lost Clients Ratio: 68.97% (20/29)
Lost clients: ['10.1.1.7', '10.1.1.9', '10.1.1.11', '10.1.1.12', '10.1.1.13', '10.1.1.15', '10.1.1.16', '10.1.1.17', '10.1.1.19', '10.1.1.20', '10.1.1.21', '10.1.1
.22', '10.1.1.23', '10.1.1.24', '10.1.1.25', '10.1.1.26', '10.1.1.27', '10.1.1.28', '10.1.1.29', '10.1.1.30']
```

Figure 8. The ratio of lost clients when the total of nodes ranges from 2 to 30

The CSMA/CA protocol is used in an ad-hoc Wi-Fi network to prevent collisions between nodes that are trying to transmit data simultaneously. Nonetheless, collisions are still possible without the RTS/CTS mechanism, which can reduce network performance, particularly as the number of nodes within a communication range grows.

V. CONCLUSION

Explain the results: Data collision is more likely to occur in ad-hoc wireless networks if all devices are linked to one another if RTS/CTS is disabled. In light of this, as the number of nodes rises, so do the number of clients and the volume of packets being sent. More collisions occur, and more packets are lost as a result.

References:

1. "Comparative Analysis of Infrastructure and Ad-Hoc Wireless Networks." *ITM Web of Conferences*, https://www.itm-conferences.org/articles/itmconf/pdf/2019/02/itmconf_icicci2018_01009.pdf.
2. "CSMA/CD and CSMA/CA Explained." *Computer Networking Notes*, 9 April 2022, https://www.computernetworkingnotes.com/networking-tutorials/csma-cd-and-csma-ca-explained.html.
3. "Flow Monitor — Model Library." *NS-3*, 21 March 2023, https://www.nsnam.org/docs/models/html/flow-monitor.html.
4. "Enable RTS/CTS?" *Google*, https://groups.google.com/g/ns-3-users/c/18BuTqRWBTs?pli=1.
5. "wifi-adhoc.cc" *NS-3*, https://www.nsnam.org/docs/release/3.19/doxygen/wifi-adhoc_8cc_source.html.
6. "Flying Adhoc Network using NS3" *LinkedIn*, https://www.linkedin.com/pulse/flying-adhoc-network-using-ns3-phd-srmieee-acmdsp/.
7. "csma-ca-simulation" *Github*, https://github.com/mindt102/csma-ca-simulation.