Labwork 5: A simulation scenario
Lecturer: Nguyen Minh Huong

Nguyen Quang Anh

Apr 9, 2024

Problem

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

Contents

1 Introduction 4

2 Design 5

2.1 Review . . . 5

2.1.1 Ad-hoc mode . . . 5

2.1.2 CSMA/CA . . . 5

2.1.3 RTS/CTS . . . 5

2.2 Protocol Description . . . 6

2.3 Scenario Design . . . 6

3 Implementing 7

4 Data Collection and Reports 8

4.1 Data Collection . . . 8

4.2 Reports . . . 8

4.2.1 Running Simulation . . . 8

4.2.2 Design the network topology . . . 9

4.2.3 Setup Applications . . . 11

4.2.4 To analyze . . . 12

4.2.5 To summarize . . . 13

4.2.6 To plot the summarized data . . . 14

5 Conclusion 16

List of Figures

2.1 CSMA/CA algorithm 6

3.1 Implementation of RTS/CTS 7

4.1 An example of a design with 7 nodes 8
4.2 Disable RTS/CTS 9
4.3 Create nodes and install Wifi model 10
4.4 Install the mobility model 10
4.5 Install Internet Stack 11
4.6 Change parameters 11
4.7 Edit directory 12
4.8 Analyze 2 nodes 13
4.9 Analyze 30 nodes 13
4.10 Analyze all XML Files 14
4.11 Plot the visualization summarized data 15

Chapter 1

Introduction

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 final project, we will use NS-3.39 to simulate an ad-hoc wireless network utilizing the CSMA/CA protocol without RTS/CTS to determine the efficiency of RTS/CTS.

Chapter 2

Design

2.1 Review

2.1.1 Ad-hoc mode

Ad-hoc mode: A wireless network structure where devices can communicate directly with each other. It allows wireless devices to form a network on the fly without the need for an existing network infrastructure.[4]
Also called peer-to-peer mode. (Decentralized, No access point)

2.1.2 CSMA/CA

It is a transmission access mechanism in a wireless network environment

2.1.3 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.

Carrier-sense multiple access with collision avoidance in computer networking where multiple network access method in which carrier sensing is used nodes attempt to avoid collisions by beginning transmission only after the channel is sensed to be ”idle”[1]

CSMA/CA is a protocol that operates in the data link layer of the OSI model

2.2 Protocol Description

The protocol being evaluated in this labwork is the CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) protocol, which is commonly used in Wifi networks operating in ad-hoc mode. CSMA/CA is a medium access control (MAC) protocol that allows multiple nodes to share a single communication channel without explicit coordination.

2.3 Scenario Design

The labwork requires evaluating the performance of the CSMA/CA protocol without the RTS/CTS (Request-to-Send/Clear-to-Send) scheme. The number of nodes within the communication range is to be increased from 2 to 30 nodes.

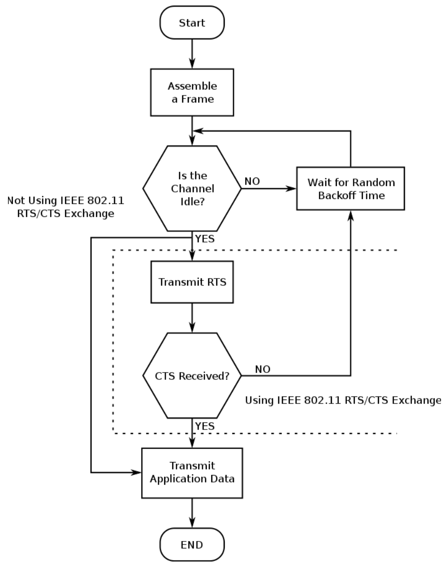

Start
Assemble a Frame
Is the Channel Idle?
NO
Wait for Random Backoff Time
Not Using IEEE 802.11 RTS/CTS Exchange
YES
Transmit RTS
CTS Received?
NO
Using IEEE 802.11 RTS/CTS Exchange
YES
Transmit Application Data
END

Figure 2.1: CSMA/CA algorithm

Chapter 3

Implementing

CSMA/CA can optionally be supplemented by the exchange of a request to send (RTS) packet sent by the sendir S and clear to send CTS packet sent by the intended receiver R. All nodes within range of the sender, receiver or both, to not transmit for the duration of the main transmission. Implementation of RTS/CTS help to partially solve the hidden node problem that is often found in wireless nether.[2]

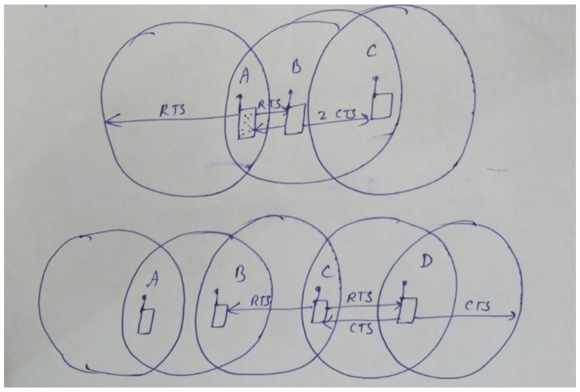


Figure 3.1: Implementation of RTS/CTS

Chapter 4

Data Collection and Reports

4.1 Data Collection

- Design

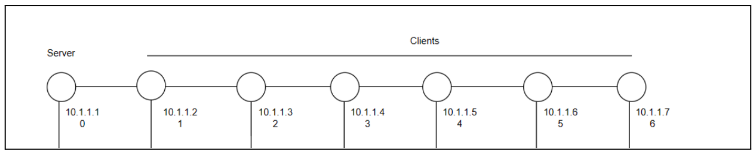


Figure 4.1: An example of a design with 7 nodes

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

4.2 Reports

4.2.1 Running Simulation

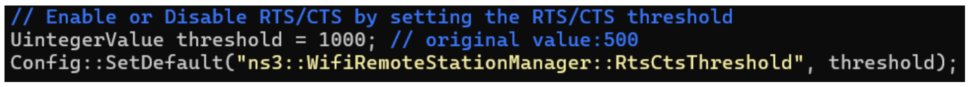
Running simulation with 2-30 nodes

```
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ../build/scratch/ns3.39-main-default
```

```
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...
```

4.2.2 Design the network topology

- Disable RTS/CTS

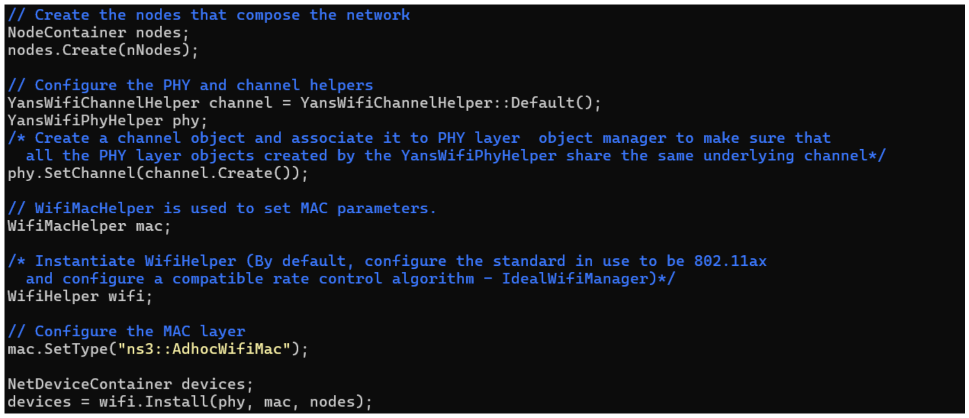


```
// Enable or Disable RTS/CTS by setting the RTS/CTS threshold
UintegerValue threshold = 1000; // original value:500
Config::SetDefault("ns3::WifiRemoteStationManager::RtsCtsThreshold", threshold);
```

Figure 4.2: Disable RTS/CTS

⇒ If packet size is less than 1000 bytes then RTS/CTS will be disabled[3]

- Create nodes and install Wifi model

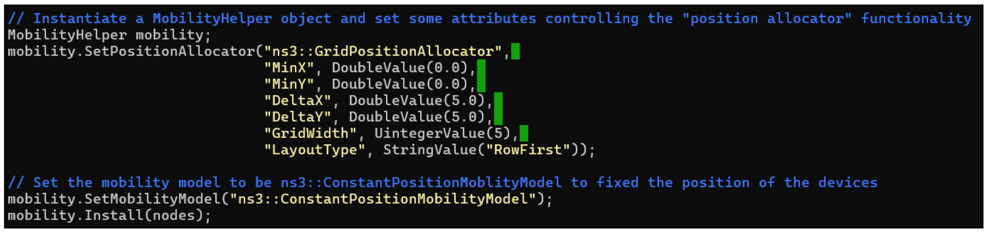


```
// Create the nodes that compose the network
NodeContainer nodes;
nodes.Create(nNodes);

// Configure the PHY and channel helpers
YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
YansWifiPhyHelper phy;
/* Create a channel object and associate it to PHY layer  object manager to make sure that
  all the PHY layer objects created by the YansWifiPhyHelper share the same underlying channel*/
phy.SetChannel(channel.Create());

// WifiMacHelper is used to set MAC parameters.
WifiMacHelper mac;

/* Instantiate WifiHelper (By default, configure the standard in use to be 802.11ax
  and configure a compatible rate control algorithm - IdealWifiManager)*/
WifiHelper wifi;

// Configure the MAC layer
mac.SetType("ns3::AdhocWifiMac");

NetDeviceContainer devices;
devices = wifi.Install(phy, mac, nodes);
```

Figure 4.3: Create nodes and install Wifi model

⇒ Configure MAC layer using ”ns3::AdhocWifiMac” to make the network works in ad-hoc mode[6]

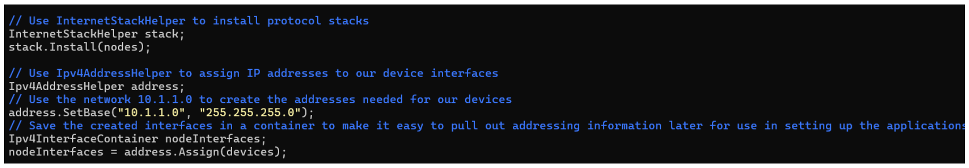
- Install the mobility model



```
// Instantiate a MobilityHelper object and set some attributes controlling the "position allocator" functionality
MobilityHelper mobility;
mobility.SetPositionAllocator("ns3::GridPositionAllocator",
                              "MinX", DoubleValue(0.0),
                              "MinY", DoubleValue(0.0),
                              "DeltaX", DoubleValue(5.0),
                              "DeltaY", DoubleValue(5.0),
                              "GridWidth", UintegerValue(5),
                              "LayoutType", StringValue("RowFirst"));

// Set the mobility model to be ns3::ConstantPositionMoblityModel to fixed the position of the devices
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(nodes);
```

Figure 4.4: Install the mobility model

- Install Internet Stack

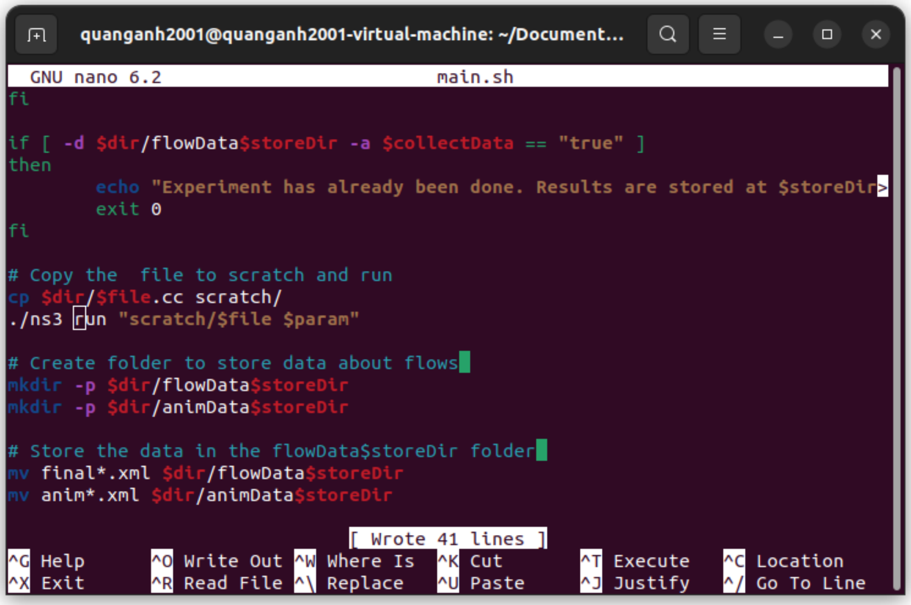


```
// Use InternetStackHelper to install protocol stacks
InternetStackHelper stack;
stack.Install(nodes);

// Use Ipv4AddressHelper to assign IP addresses to our device interfaces
Ipv4AddressHelper address;
// Use the network 10.1.1.0 to create the addresses needed for our devices
address.SetBase("10.1.1.0", "255.255.255.0");
// Save the created interfaces in a container to make it easy to pull out addressing information later for use in setting up the applications
Ipv4InterfaceContainer nodeInterfaces;
nodeInterfaces = address.Assign(devices);
```

Figure 4.5: Install Internet Stack

4.2.3 Setup Applications

1. cd into folder ”ns-3.39”
2. Clone this repository[5]
3. Move main.sh from NSFinalProject into ns-3.34: NSFinalProject/main.sh
4. Edit main.sh to change the parameters

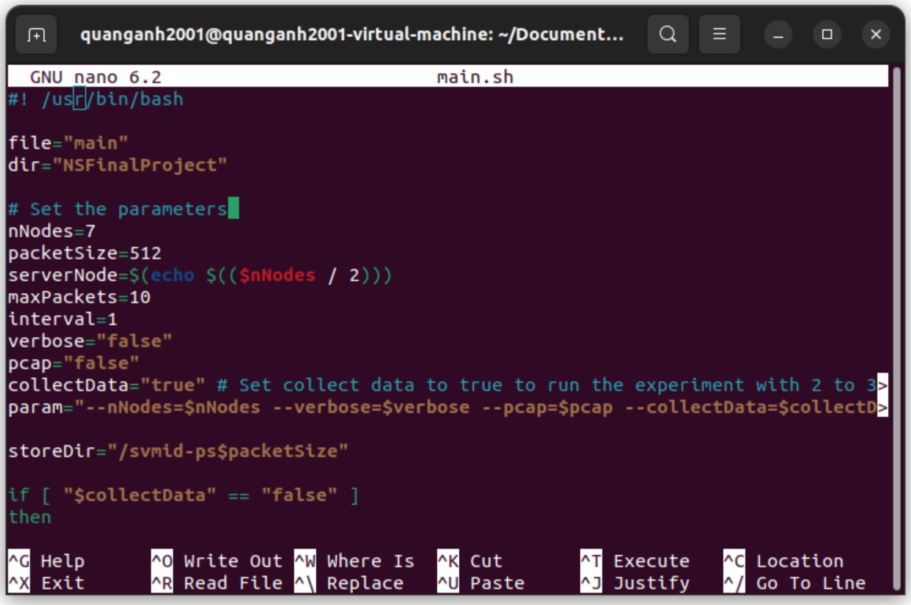


quanganh2001@quanganh2001-virtual-machine: ~/Document...

```
  GNU nano 6.2                                main.sh
fi

if [ -d $dir/flowData$storeDir -a $collectData == "true" ]
then
        echo "Experiment has already been done. Results are stored at $storeDir>
        exit 0
fi

# Copy the  file to scratch and run
cp $dir/$file.cc scratch/
./ns3 run "scratch/$file $param"

# Create folder to store data about flows
mkdir -p $dir/flowData$storeDir
mkdir -p $dir/animData$storeDir

# Store the data in the flowData$storeDir folder
mv final*.xml $dir/flowData$storeDir
mv anim*.xml $dir/animData$storeDir

                                [ Wrote 41 lines ]
^G Help         ^O Write Out  ^W Where Is   ^K Cut         ^T Execute     ^C Location
^X Exit         ^R Read File  ^\ Replace    ^U Paste       ^J Justify     ^/ Go To Line
```

Figure 4.6: Change parameters

5. Run main.sh: ./main.sh



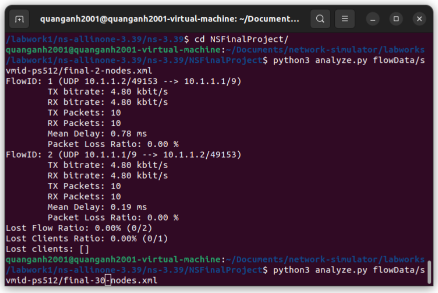
```
  GNU nano 6.2                              main.sh
#! /usr/bin/bash

file="main"
dir="NSFinalProject"

# Set the parameters
nNodes=7
packetSize=512
serverNode=$(echo $(($nNodes / 2)))
maxPackets=10
interval=1
verbose="false"
pcap="false"
collectData="true" # Set collect data to true to run the experiment with 2 to 3>
param="--nNodes=$nNodes --verbose=$verbose --pcap=$pcap --collectData=$collectD>

storeDir="/svmid-ps$packetSize"

if [ "$collectData" == "false" ]
then

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

Figure 4.7: Edit directory

4.2.4 To analyze

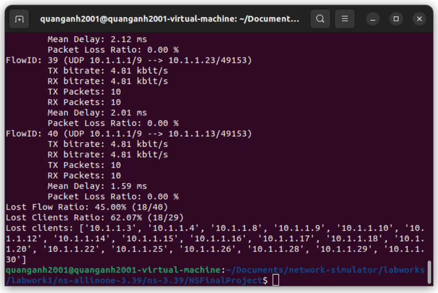
1. cd into NSFinalProject
2. To analyze an XML file: python3 analyze.py flowData/svmid-ps512/final-2-nodes.xml
3. To analyze all XML file after running with 2 to 30 nodes: ./analyzeAll.sh svmid-ps512

After analyzing data is summarized automatically and save at *analyzedData/"configuration"* and *summarizedData/"configuration"* e.g.: analyzedData/svmid-ps512



```
/labwork1/ns-allinone-3.39/ns-3.39$ cd NSFinalProject/
quanganh2001@quanganh2001-virtual-machine:~/Documents/network-simulator/labworks
/labwork1/ns-allinone-3.39/ns-3.39/NSFinalProject$ python3 analyze.py flowData/s
vmid-ps512/final-2-nodes.xml
FlowID: 1 (UDP 10.1.1.2/49153 --> 10.1.1.1/9)
        TX bitrate: 4.80 kbit/s
        RX bitrate: 4.80 kbit/s
        TX Packets: 10
        RX Packets: 10
        Mean Delay: 0.78 ms
        Packet Loss Ratio: 0.00 %
FlowID: 2 (UDP 10.1.1.1/9 --> 10.1.1.2/49153)
        TX bitrate: 4.80 kbit/s
        RX bitrate: 4.80 kbit/s
        TX Packets: 10
        RX Packets: 10
        Mean Delay: 0.19 ms
        Packet Loss Ratio: 0.00 %
Lost Flow Ratio: 0.00% (0/2)
Lost Clients Ratio: 0.00% (0/1)
Lost clients: []
quanganh2001@quanganh2001-virtual-machine:~/Documents/network-simulator/labworks
/labwork1/ns-allinone-3.39/ns-3.39/NSFinalProject$ python3 analyze.py flowData/s
vmid-ps512/final-30-nodes.xml
```

Figure 4.8: Analyze 2 nodes



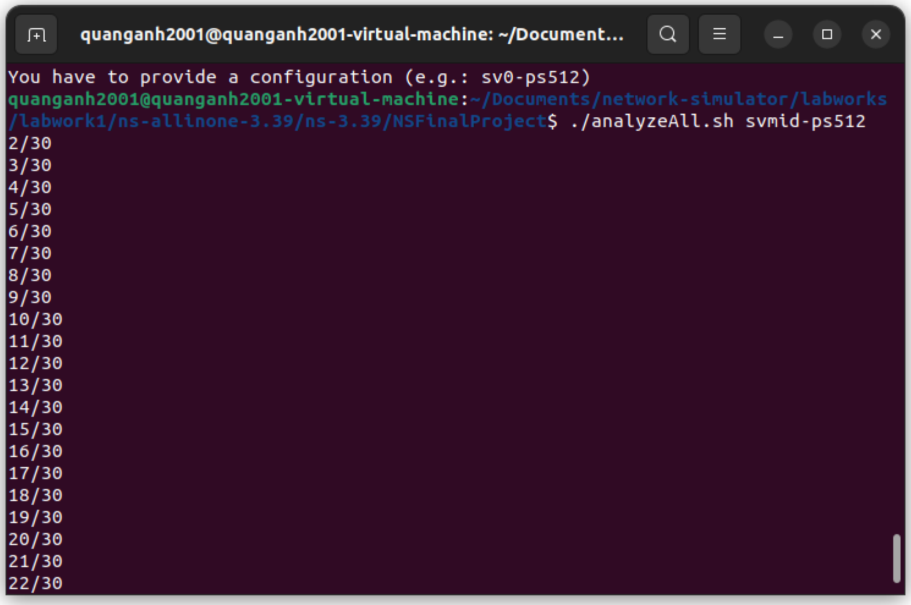
```
        Mean Delay: 2.12 ms
        Packet Loss Ratio: 0.00 %
FlowID: 39 (UDP 10.1.1.1/9 --> 10.1.1.23/49153)
        TX bitrate: 4.81 kbit/s
        RX bitrate: 4.81 kbit/s
        TX Packets: 10
        RX Packets: 10
        Mean Delay: 2.01 ms
        Packet Loss Ratio: 0.00 %
FlowID: 40 (UDP 10.1.1.1/9 --> 10.1.1.13/49153)
        TX bitrate: 4.81 kbit/s
        RX bitrate: 4.81 kbit/s
        TX Packets: 10
        RX Packets: 10
        Mean Delay: 1.59 ms
        Packet Loss Ratio: 0.00 %
Lost Flow Ratio: 45.00% (18/40)
Lost Clients Ratio: 62.07% (18/29)
Lost clients: ['10.1.1.3', '10.1.1.4', '10.1.1.8', '10.1.1.9', '10.1.1.10', '10.
1.1.12', '10.1.1.14', '10.1.1.15', '10.1.1.16', '10.1.1.17', '10.1.1.18', '10.1.
1.20', '10.1.1.22', '10.1.1.25', '10.1.1.26', '10.1.1.28', '10.1.1.29', '10.1.1.
30']
quanganh2001@quanganh2001-virtual-machine:~/Documents/network-simulator/labworks
/labwork1/ns-allinone-3.39/ns-3.39/NSFinalProject$ 
```

Figure 4.9: Analyze 30 nodes

- *Data from Flow Monitor*: Data is stored in flows that contain all data about packets sent by a particular host to another[7]
- Data from the flows contains a lot of interesting information such as
  - The time the first and last packet is transmitted and received
  - The total delay
  - The total bytes and packets transmitted and received
  - Number of lost packets

4.2.5 To summarize

1. cd into NSFinalProject: cd NSFinalProject
2. Run the summarize script to see loss flow ratio of each case: ./summarize.sh ”configuration” or bash summarize.sh ”configuration”. (e.g.: ./summarize.sh svmid-ps512



```
quanganh2001@quanganh2001-virtual-machine: ~/Document...
You have to provide a configuration (e.g.: sv0-ps512)
quanganh2001@quanganh2001-virtual-machine:~/Documents/network-simulator/labworks
/labwork1/ns-allinone-3.39/ns-3.39/NSFinalProject$ ./analyzeAll.sh svmid-ps512
2/30
3/30
4/30
5/30
6/30
7/30
8/30
9/30
10/30
11/30
12/30
13/30
14/30
15/30
16/30
17/30
18/30
19/30
20/30
21/30
22/30
```

Figure 4.10: Analyze all XML Files

4.2.6 To plot the summarized data

1. cd into NSFinalProject
2. Run the python script: python3 plot.py summarizedData/svmid-ps512.csv. Using matplotlib to visualize the lost clients ratio.

When the number of nodes increases, the ratio of lost client also increases
The CSMA/CA protocol is used in an ad-hoc Wifi 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.

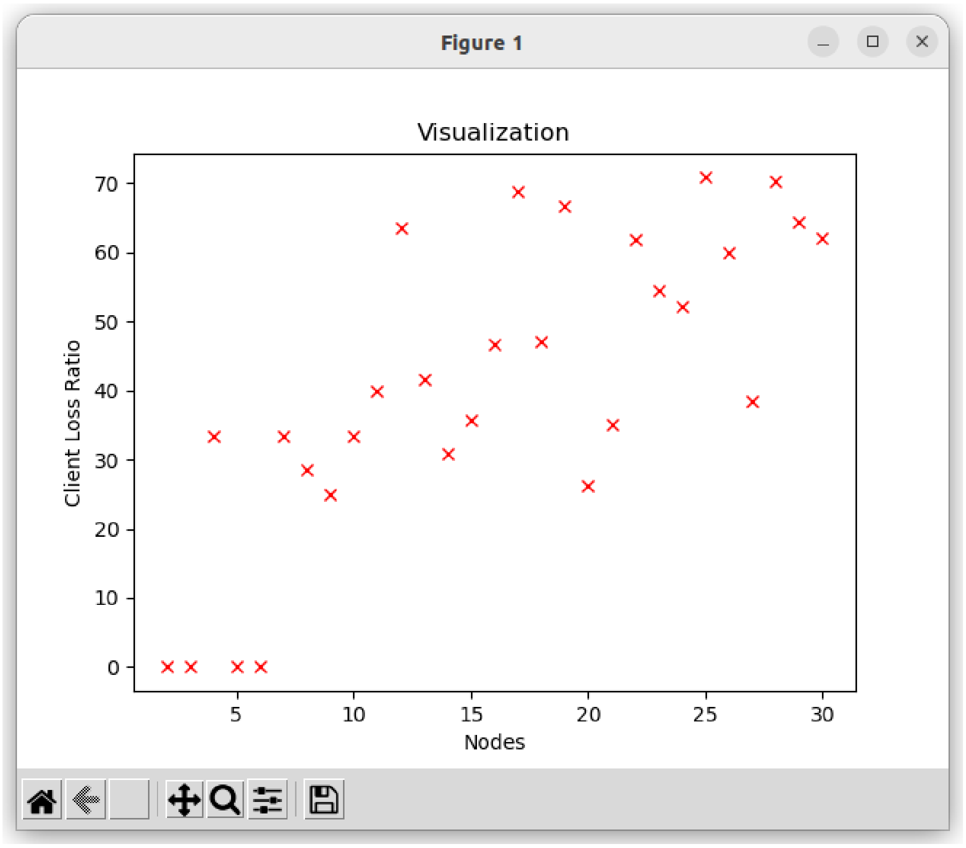


Figure 4.11: Plot the visualization summarized data

Chapter 5

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.

Bibliography

[1] bartleby. Problem: Consider csma/ca protocol in wifi networks that working in ad-hoc mode... `https://www.bartleby.com/questions-and-answers/problem-consider-csmaca-protocol-in-wifi-networks-that-working-in-ad-hoc-mode.-evaluate-performance-/6935c973-0b88-40a1-9b4c-e308061162db`, Last accessed on 2024-04-09.

[2] Computer Networking Notes. Csma/cd and csma/ca explained, 2024. `https://www.computernetworkingnotes.com/networking-tutorials/csma-cd-and-csma-ca-explained.html`, Last accessed on 2024-04-07.

[3] Google. Enable rts/cts?, 2012. `https://groups.google.com/g/ns-3-users/c/18BuTqRWBTs?pli=1`, Posted on 30 October 2012.

[4] Shanna Li. Comparative analysis of infrastructure and ad-hoc wireless networks. In *ITM Web of Conferences*, volume 25, page 01009. EDP Sciences, 2019.

[5] mindt102. Csma-ca simulation, 2022. `https://github.com/mindt102/csma-ca-simulation`, Last updated 2023-11-10.

[6] nsnam. examples/wireless/wifi-adhoc.cc, 2014. `https://www.nsnam.org/docs/release/3.19/doxygen/wifi-adhoc_8cc_source.html`, Generated on April 19, 2014.

[7] nsnam. Flow monitor, 2024. `https://www.nsnam.org/docs/models/html/flow-monitor.html`, Last updated 2024-04-08.