Wireless Sensor Network Design Document

Project Name: S-MAC Simulation

Faculty: School of Electronic Science and Engineering

Advisor: Prof. Qian Tian

Name: Yuan Lei

Data: Dec. 2020

(The document is machine translated from Chinese. Though I've done some proofreading, hope there's no more big typo or grammar issue. Thx for reading :p)

I, the experimental requirements:

- Establish a wireless sensing network
  - The S-Mac protocol is used in the MAC layer
  - The built-in network protocol or REACH protocol is used at the network layer
  - Transport layer protocol optional
  - The topology is optional, and the number of nodes is not less than eight
  - Send data customizations
- Use NS2 for network simulation
- Use the NAM tool to graphically display the simulation results
- Analyze simulation results and write technical reports

II, the experimental content:

1. The network-related custom parameters are as follows:

- Number of nodes: 8
- Topology: Y type
- Channel: Wireless channel
- Propagation model: TwoRayGround
- Physical layer type: WirelessPhy
- MAC protocol: S-MAC/802.11
- Routing Protocol: DSR (My environment uses the AODV protocol and there will be a system segment error memory overflow, I don't know why, I can only change it.) DSR)
- Transport Layer Protocol: UDP
- Queue type: DropTail/PriQueue
- Packet size: 500Byte
- Energy Model: EnergyModel
- Initial energy: 500J/ time unit
  - ✓ Idle energy: 1J / time unit
  - ✓ Sleep energy: 0.01J / time unit
  - ✓ Transmitted energy: 0.5J/time unit
  - ✓ Received energy: 1J/time unit
  - ✓ Transmit energy: 1J/time unit

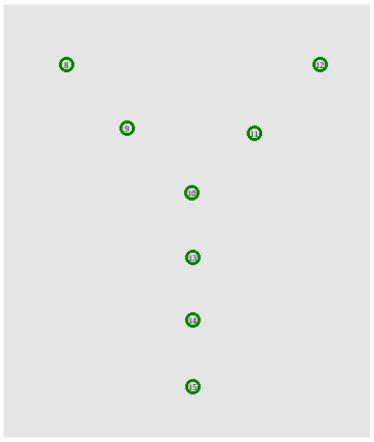


Figure 1. Y shape topology network.

2. Brief summary of S-MAC (Sensor Media Access Control) protocol

S-MAC is based on the CSMA/CA mechanism, which belongs to the competitive MAC protocol, and introduces cycle sleep and cycle listening mechanisms to reduce idle listening and virtual cluster mechanisms 。 Nodes within the same virtual cluster in the network synchronize using the same sleep and wake modes, all operating at the same duty cycle. Use a synchronous frame sending mechanism to ensure that all nodes in the cluster wake up and sleep at the same time. When the nodes are in a wake-up state, they listen, and according to the listening results, they determine whether to send or receive data, and when all nodes are asleep, the RF transceiver is automatically turned off to save energy. The main advantage of the S-MAC protocol is that it can make the nodes go to sleep periodically, saving energy consumption, and because of the existence of a virtual cluster mechanism, it is not necessary for all nodes to be synchronized. The drawback is the introduction of sleep and increased network latency. The timing of listening and sleeping is also fixed and does not dynamically adjust as the flow changes. In addition, the message sending mechanism also makes it less fair.

Timing diagram of S-MAC protocol:

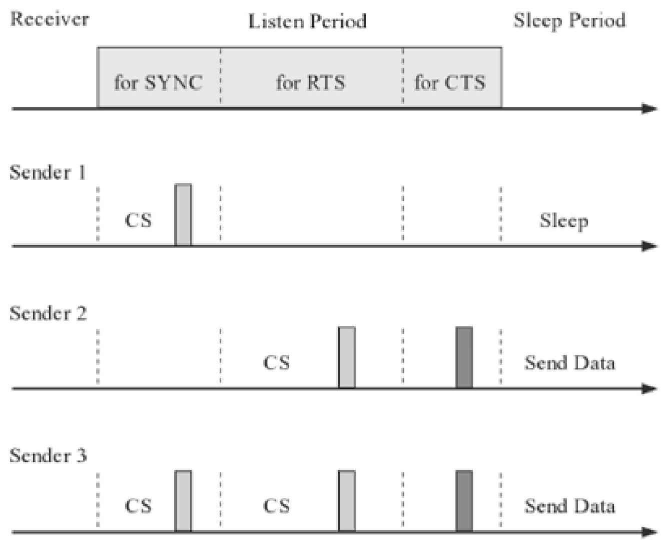


Figure 2. time diagram of S-MAC protocol

3. Purpose of the experiment

Under the custom parameters, the differences between the S-MAC protocol and the 802.11 protocol in terms of latency, packet loss rate, energy consumption, and throughput are analyzed.

III, the experimental protocol

1. Build the network model (gradually established from upper to lower levels).

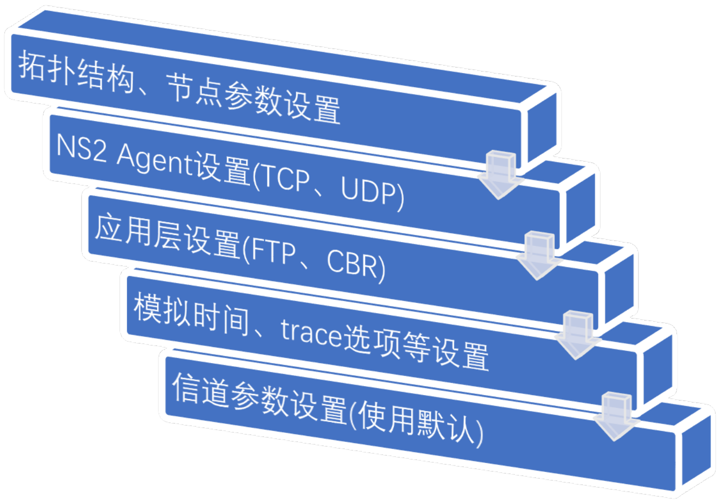


Figure 3. Process of establishing the network

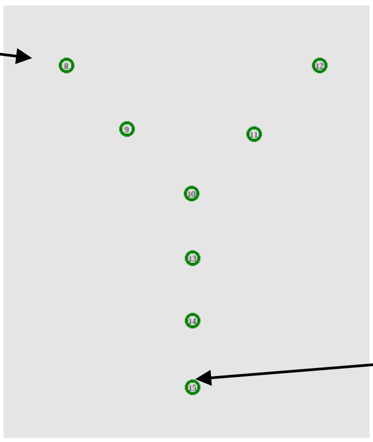
Building Results:

Source Node: The UDP Agent that binds the traffic generator

CBR bitrate parameter:

The packet size is 500 Bytes

The send interval is 0.005s

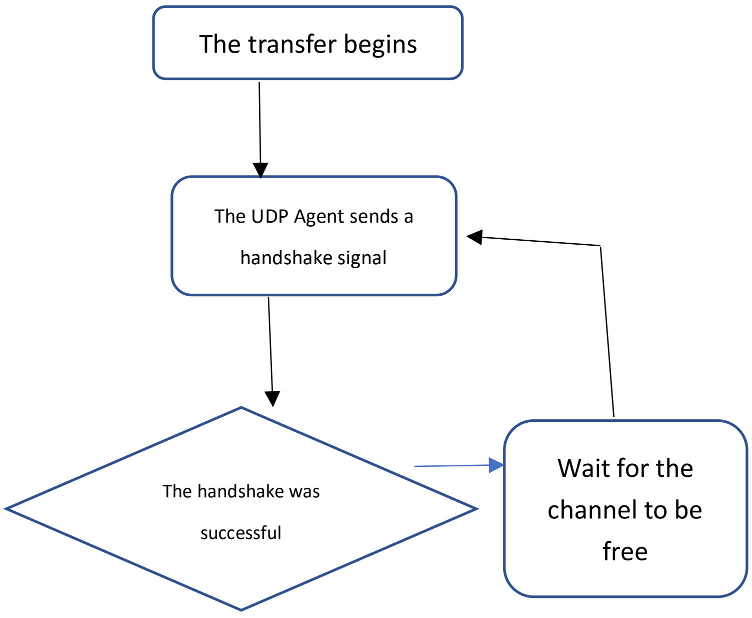


Sink nodes

Figure 4. network topology

(There may be problems with the parameter settings here, which is analyzed in the packet loss section).

2. Experimental process


The transfer begins
The UDP Agent sends a handshake signal
The handshake was successful
Wait for the channel to be free

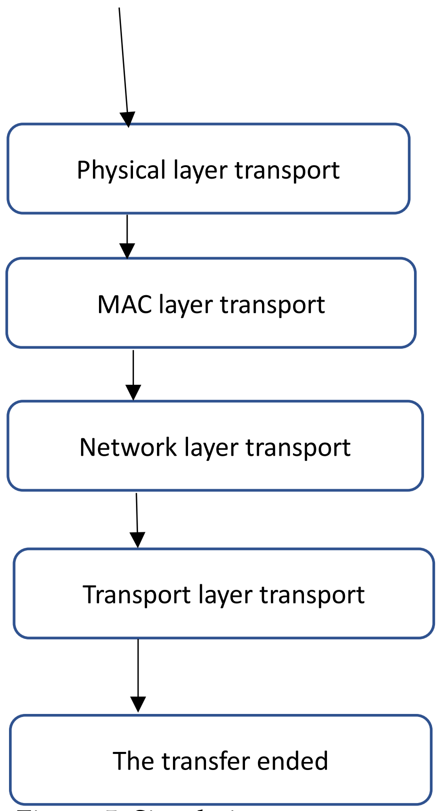
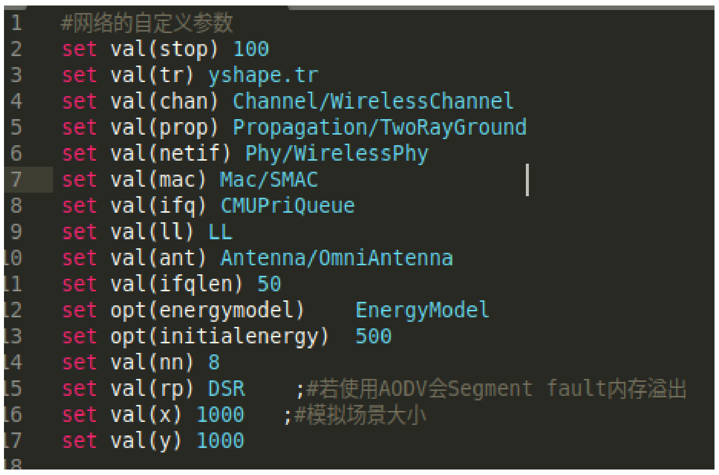


Figure 5. Simulation process

After the topology is established, bind the UDP Agent to the initial node, and establish a CBR traffic generator with a packet size of 500Bytes and a send interval 5ms, bound to the UDP of the initial node. Bind the data sink null on the aggregation node, and then connect the udp and null. The subsequent transfer operations of the various handshakes can be automatically simulated using NAM.

IV, Code Modules

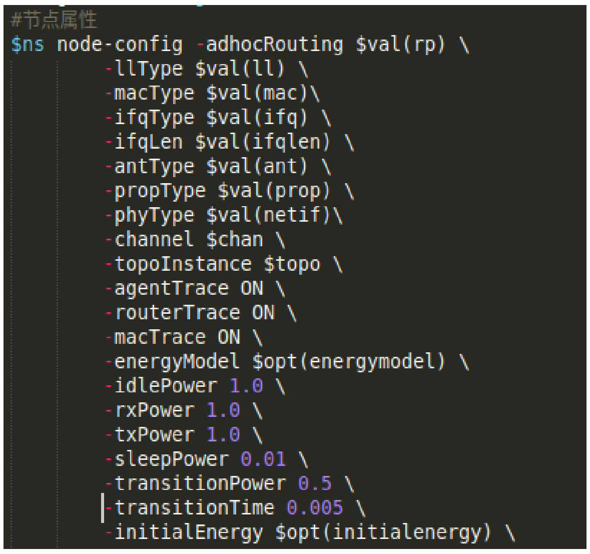
1. Network custom parameters



```
#网络的自定义参数
set val(stop) 100
set val(tr) yshape.tr
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy
set val(mac) Mac/SMAC
set val(ifq) CMUPriQueue
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(ifqlen) 50
set opt(energymodel)    EnergyModel
set opt(initialenergy)  500
set val(nn) 8
set val(rp) DSR     ;#若使用AODV会Segment fault内存溢出
set val(x) 1000    ;#模拟场景大小
set val(y) 1000
```

2. Node properties

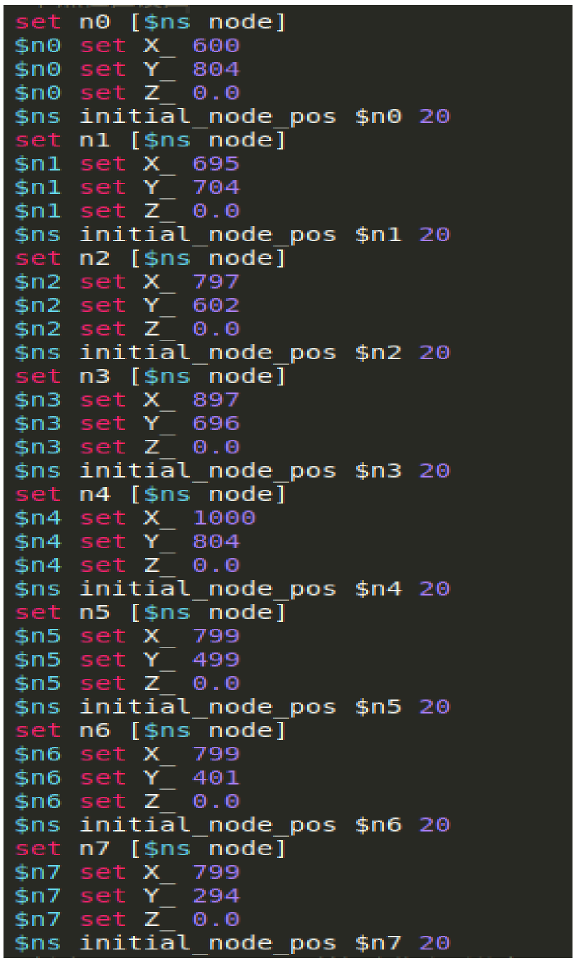
A global parameter that is set before the node property is set before the species is joined. Energy analysis can directly use the energy model of ns2 to add the relevant Power parameters at the node. (However, the introduction of Energy Model will cause the output trace file format to be inconsistent, and it will be difficult for the script to extract each piece of data separately.)



```
#节点属性
$ns node-config -adhocRouting $val(rp) \
        -llType $val(ll) \
        -macType $val(mac)\
        -ifqType $val(ifq) \
        -ifqLen $val(ifqlen) \
        -antType $val(ant) \
        -propType $val(prop) \
        -phyType $val(netif)\
        -channel $chan \
        -topoInstance $topo \
        -agentTrace ON \
        -routerTrace ON \
        -macTrace ON \
        -energyModel $opt(energymodel) \
        -idlePower 1.0 \
        -rxPower 1.0 \
        -txPower 1.0 \
        -sleepPower 0.01 \
        -transitionPower 0.5 \
        -transitionTime 0.005 \
        -initialEnergy $opt(initialenergy) \
```

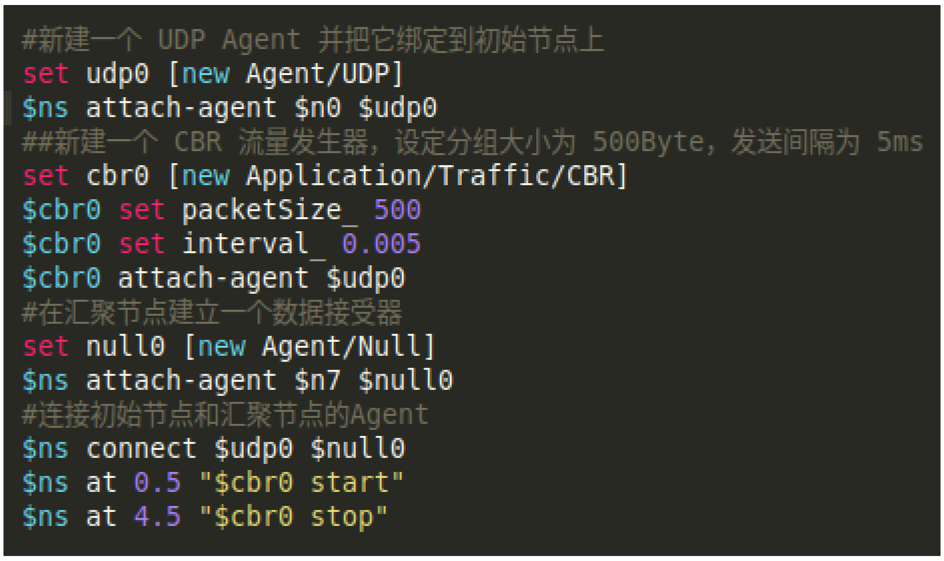
3. Topology settings

The topology is roughly set up and the shape is roughly aligned using the NAM display for fine adjustment.



```
set n0 [$ns node]
$n0 set X_ 600
$n0 set Y_ 804
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 695
$n1 set Y_ 704
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 797
$n2 set Y_ 602
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 897
$n3 set Y_ 696
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 1000
$n4 set Y_ 804
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 799
$n5 set Y_ 499
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 799
$n6 set Y_ 401
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 799
$n7 set Y_ 294
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
```

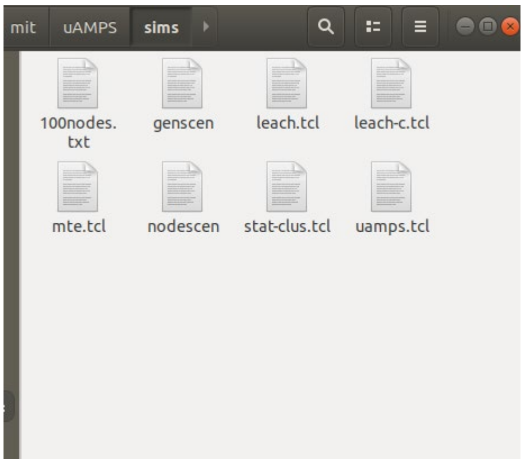
4. Transport layer and application layer settings
Agent is an important feature in ns2, mainly responsible for the establishment and description of the transport layer, mainly set up with reliable transmission tcp and unreliable transmission of UDP two, will be generator Attach to the initial node, and then attach the receiver to the aggregation node. The application layer settings use Application to set the packet parameters and then attach to the initial node, and then use connect to establish the transmission by connecting the initial node and the aggregation node through the transport layer 。


```
#新建一个 UDP Agent 并把它绑定到初始节点上
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
##新建一个 CBR 流量发生器，设定分组大小为 500Byte，发送间隔为 5ms
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
#在汇聚节点建立一个数据接受器
set null0 [new Agent/Null]
$ns attach-agent $n7 $null0
#连接初始节点和汇聚节点的Agent
$ns connect $udp0 $null0
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

5. Newly added LEACH protocol
LEACH Protocol Content Summary:
Basic idea: Randomly loop through the network cluster head, dynamically build the cluster, and distribute the energy consumption evenly throughout the network.
Specific process: the cluster head node that is randomly selected first broadcasts the information synchronously, and other nodes form a cluster according to the cluster head broadcast information they receive, and if multiple broadcast messages are received at the same time, the node with the strongest energy is selected as the cluster head 。 TDMA is then used to generate a time schedule for sending and receiving from point to cluster header. After the stabilization stage, the node will make data to the tuft hair, and the tuft head will make data fusion before forwarding the data.

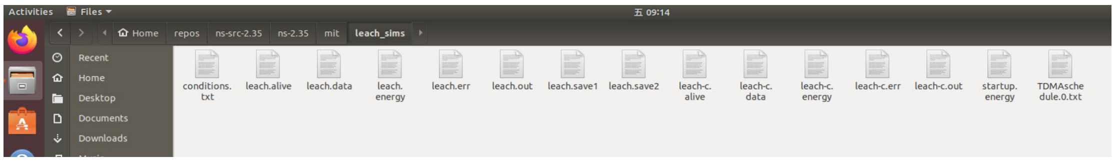
Through the reference information, it can be seen that the Massachusetts Institute of Technology (MIT) in the United States developed a mature SET OFL ON NS2 around 2000, and it is difficult to write a new protocol , so this section is mainly about learning how to add new protocols and recompile ns2

First download the leach-setup.sh and ns-leach2.35 .tar .gz from network resources https://drive.google.com/drive/folders/0B4nUSbTYSK4TfkVrVEhzaHR1aXYwQ296bXZKZE14a21wSXFtRmFqNW4tQ2VKbVlGWnJUeXM

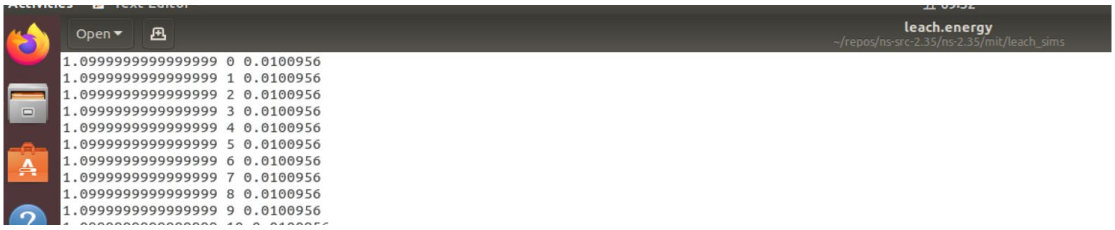


After extracting, copy the mit folder in the folder to the ns-2.35 directory. The contents of the Mit folder are as follows:
It is necessary to modify the sims/uAMPS and tcl/mobility directories within the mit folder leach-c.tcl, mte.tcl, leach. tcl、 uamps. The source option of tcl's four file headers (probably a BUG left behind by the lack of flexibility of the tcl script).

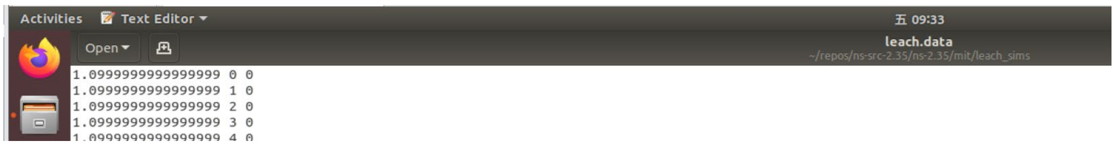
The original directory looked like this: `source /home/pradeepkumar/ns-allinone-2.35/ns-2.35/mit/`
The author of the MIT project at the beginning of the name P should change this directory of all the above files to their own corresponding directo `source /home/ray/repos/ns-src-`
Change the source of all tcl files after using Terminal to run under the ns-2.35 folder before copying leach _test
In the leach folder under the mit directory_sims there are several files as shown in the figure:



. energy and If there is a content output in the data file, it can be determined that the LEACH protocol has been added successfully.



```
1.0999999999999999 0 0.0100956
1.0999999999999999 1 0.0100956
1.0999999999999999 2 0.0100956
1.0999999999999999 3 0.0100956
1.0999999999999999 4 0.0100956
1.0999999999999999 5 0.0100956
1.0999999999999999 6 0.0100956
1.0999999999999999 7 0.0100956
1.0999999999999999 8 0.0100956
1.0999999999999999 9 0.0100956
```



```
1.0999999999999999 0 0
1.0999999999999999 1 0
1.0999999999999999 2 0
1.0999999999999999 3 0
1.0999999999999999 4 0
```

...

However, after taking a lot of effort to configure REACH, in order to be able to call and simulate REACH in tcl and nam, you need to modify the entire ns2 tcl_library. I checked different tutorials on Google and still can't write a good yshape before The reach protocol is called in the tcl file.... (Most likely, the library was messed up by me).)

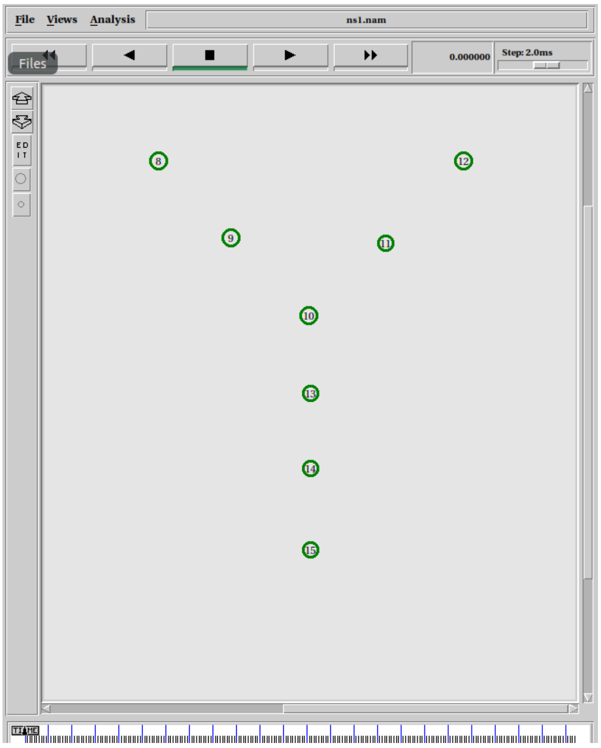
In the end, I had to give up out of desperation, and continued to improve the project after the senior year of independent time to start preparing for the graduation project, and became more familiar with the computer network system and Linux operating system. I also learned that ns2 has been replaced by a new generation of NetworkSimulator3 (ns3), which supports the use of Python, then writing protocols and network simulations in python is easier to get started with and use than ns2.

V. Experimental results and analysis

The experimental results use the SMAC protocol to compare with the MAC protocol of 802.11, and the structure and parameters are the same except for the MAC layer protocol

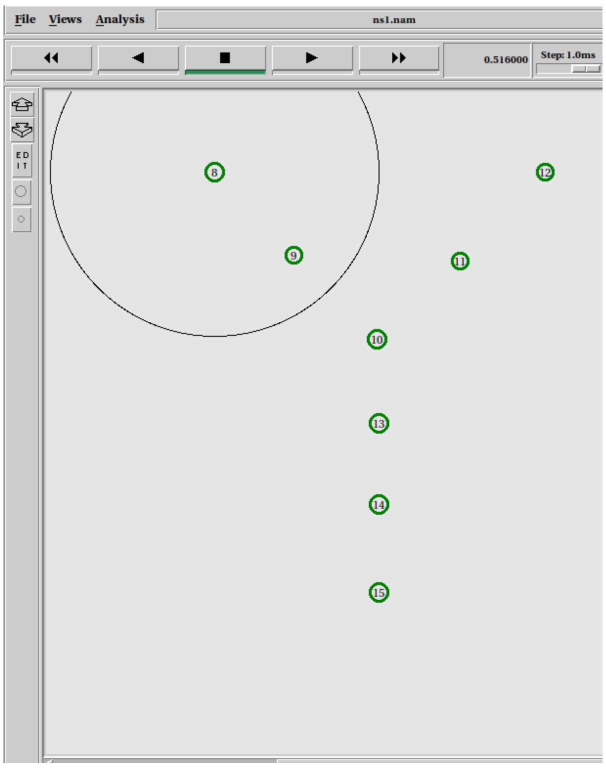
1. Topology:

Here the node label is a little problematic, the node I set is n0 ~ n7, but the node number is set there is a little strange, the label is incremented from the number of nodes 8, so n0 ~ n7 Corresponds to n8~n15 respectively

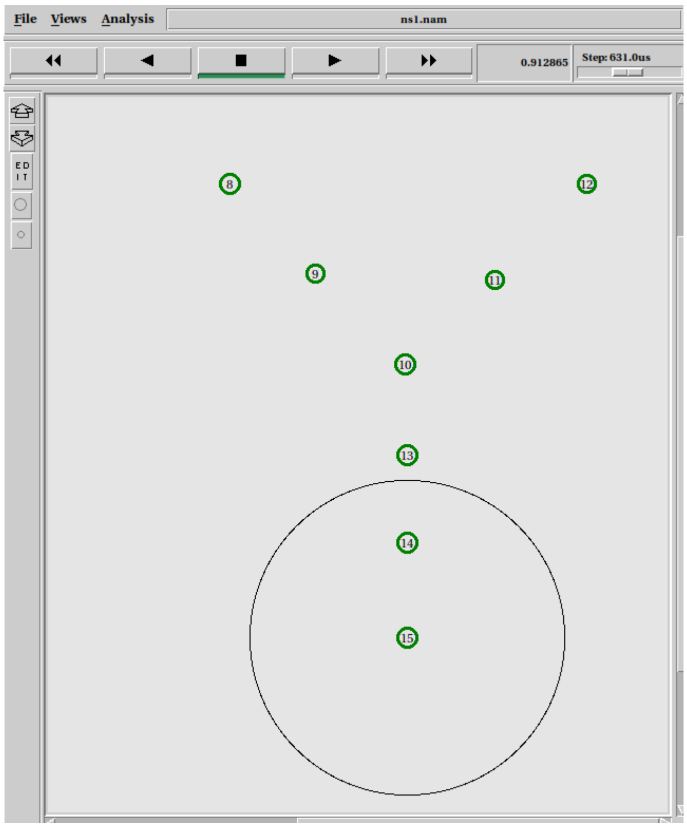


2. Data transmission request

For both SMAC and 802.11, the initial node sends the request around 0.5 seconds

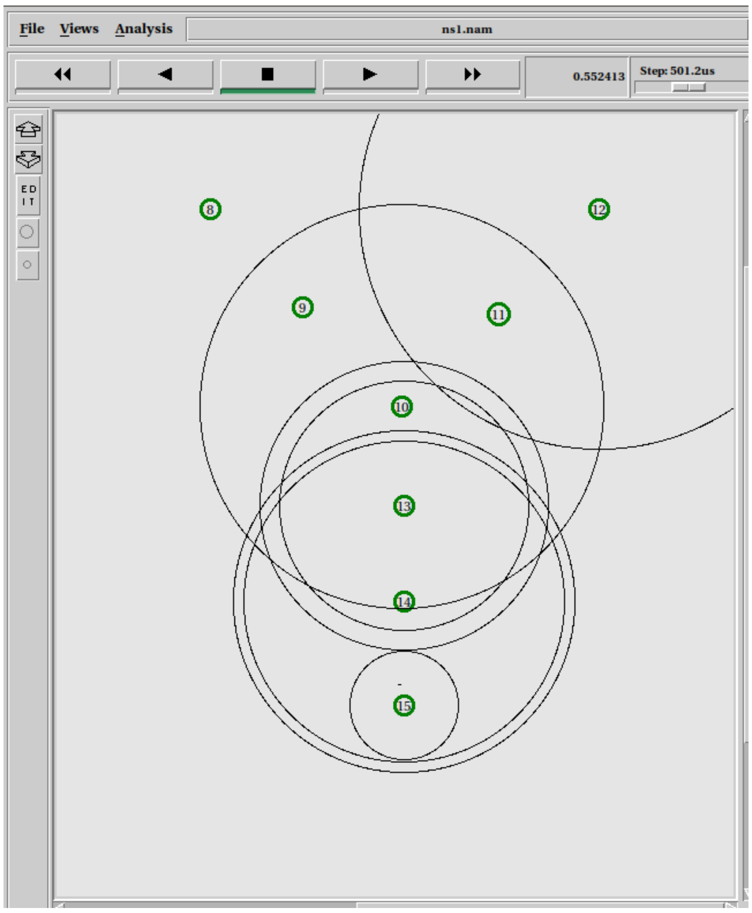


For SMAC protocol: The aggregation node receives RTS and returns CTS at about 0.9 seconds.



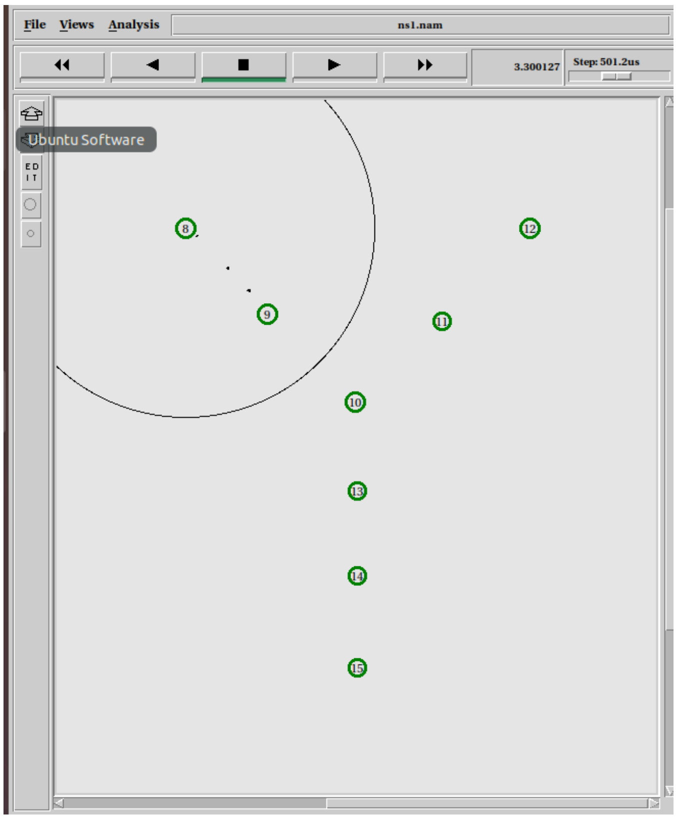
For the MAC protocol of 802.11: The aggregation node receives RTS and returns CTS at about 0.55s.

It can be seen that the response time of the 802.11 protocol is faster than that of SMAC, and the latency is lower than that of SMAC.

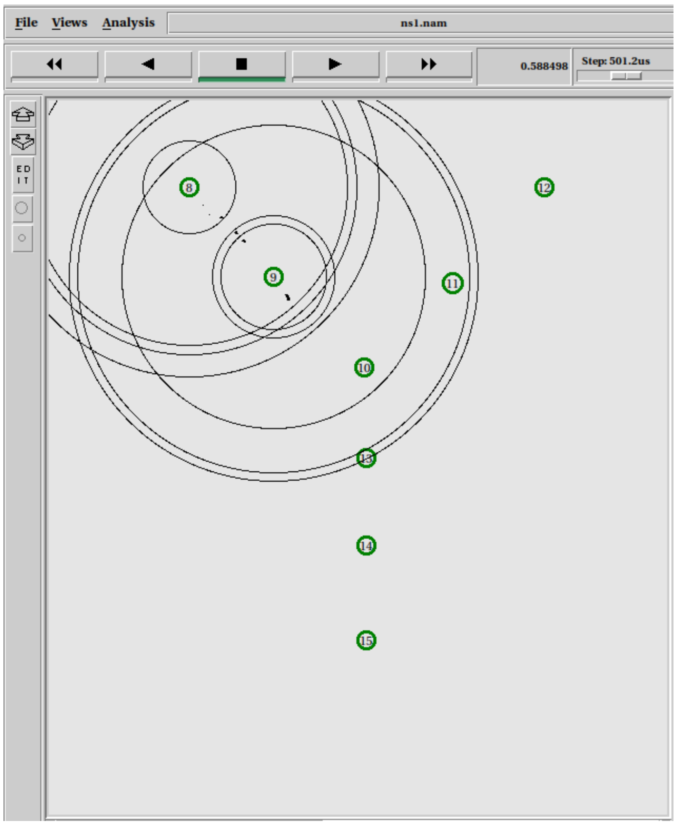


3. Data transmission

For SMAC: Start data transfer around 3.3s

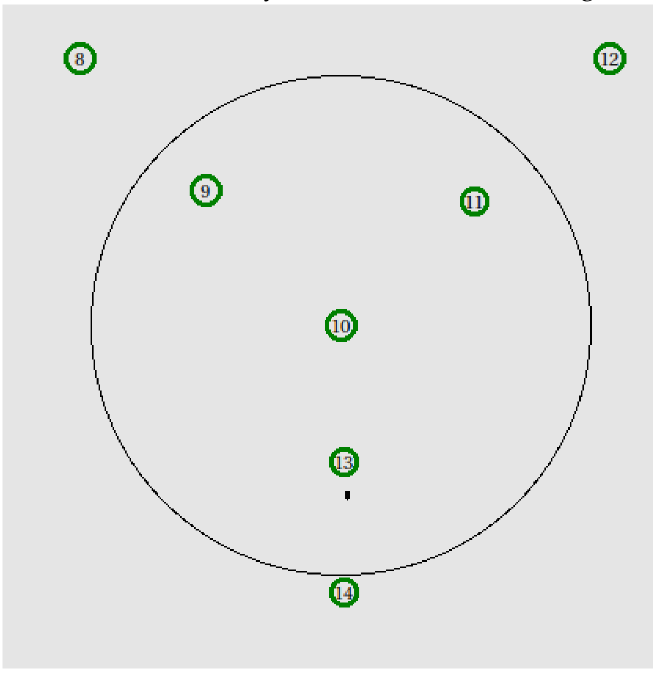


For the MAC protocol of 802.11: Data transfer starts at about 0.58s

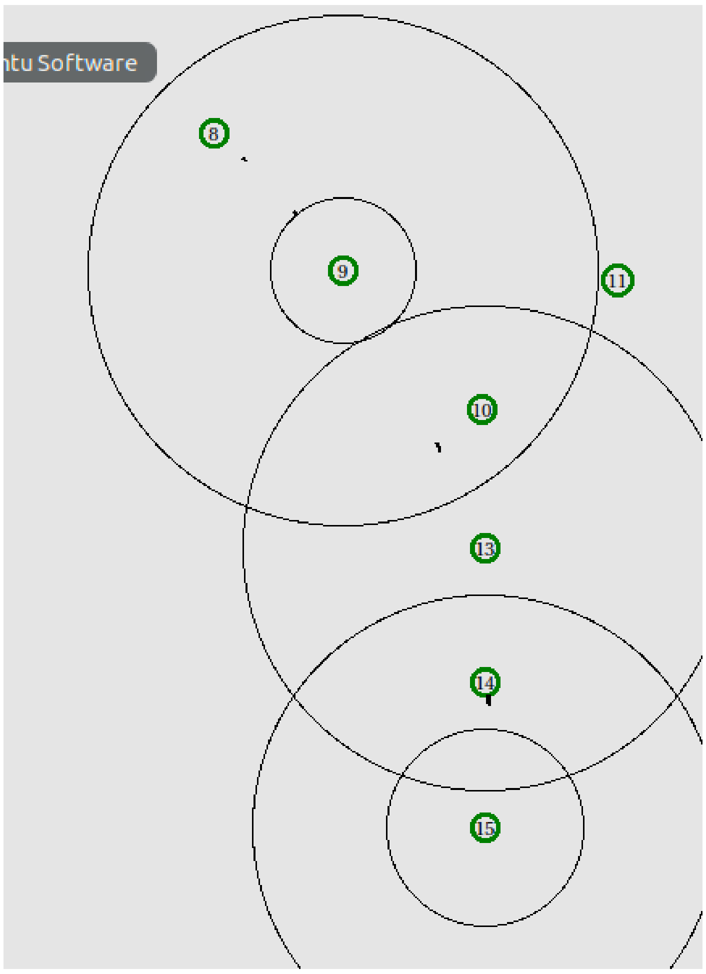


It can also be concluded that 802.11 has a faster response speed and lower latency than SMAC.

It is worth noting during the transmission of SMAC that since n6 (label 14) is in n3 (label 10) is in range, so if n6 sends a message to n3 to n6, it skips the middle n5 (label 13) Pass directly to n6, as shown in the figure below:

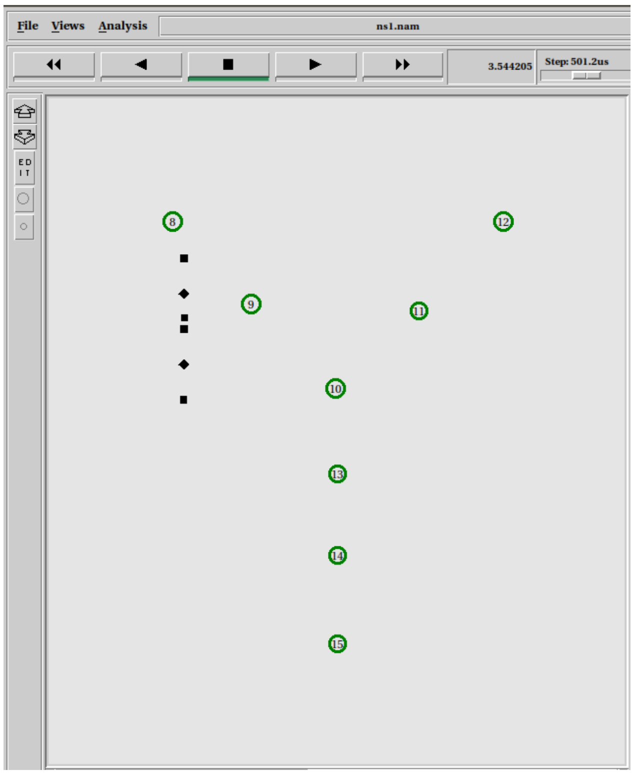


The transmission process of 802.11 is also similar, because the node of label 13 is within the listening range of label 9, and the node of label 10 is directly skipped during data transmission, which reduces the transmission delay to some extent. As shown in the following figure:

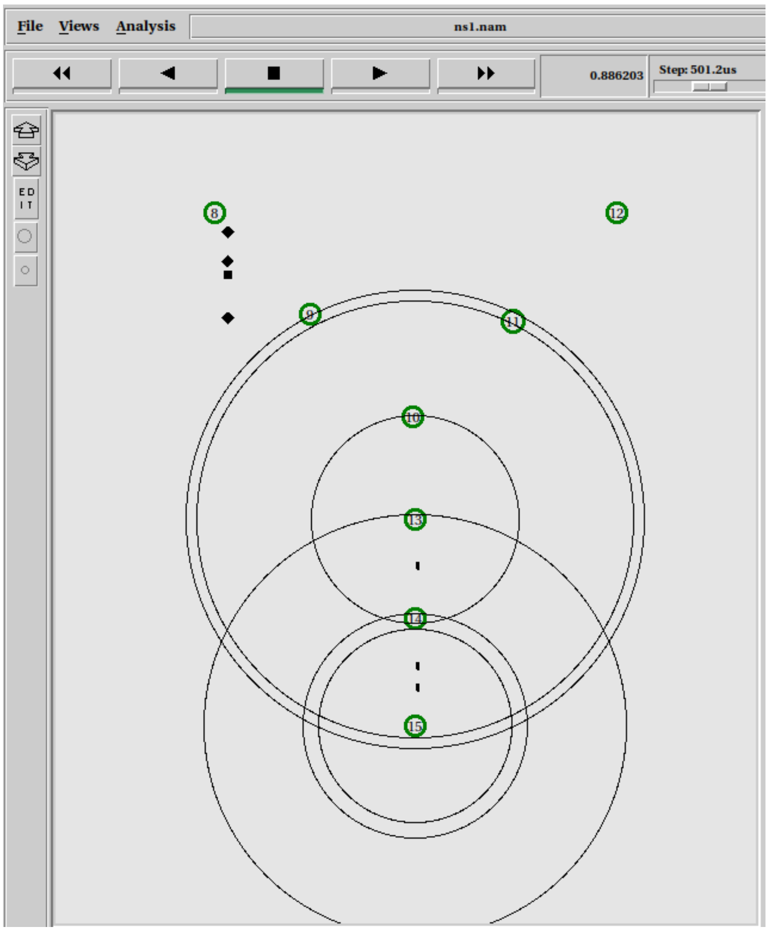


4. Packet loss

For the SMAC protocol: Packet loss starts to occur around 3.5 seconds



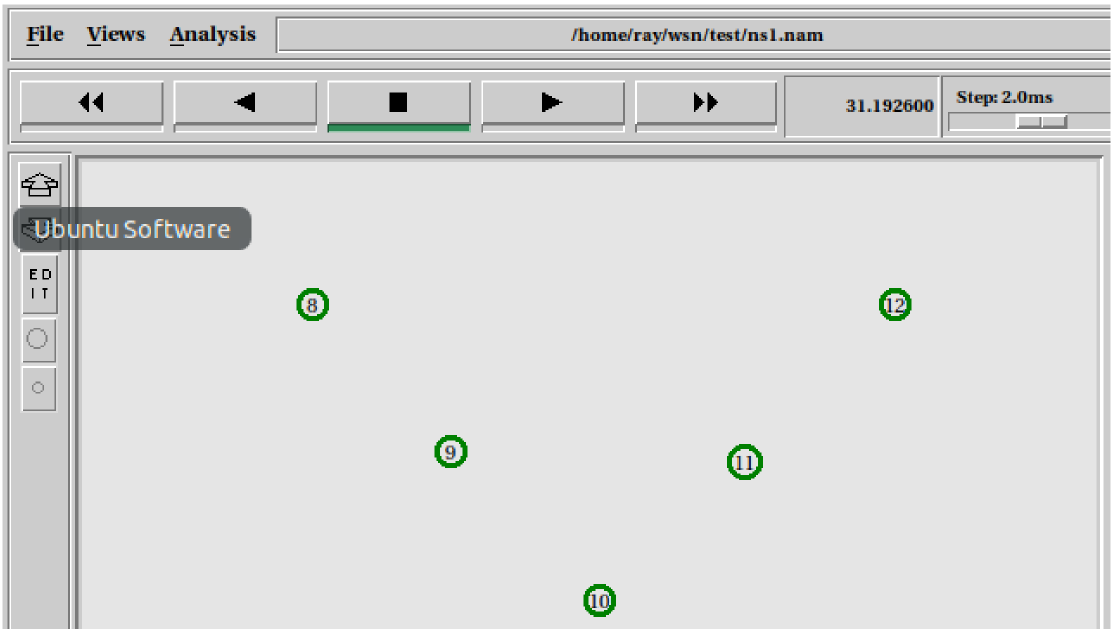
For the 802.11 MAC protocol: Packet loss begins at about 0.88s



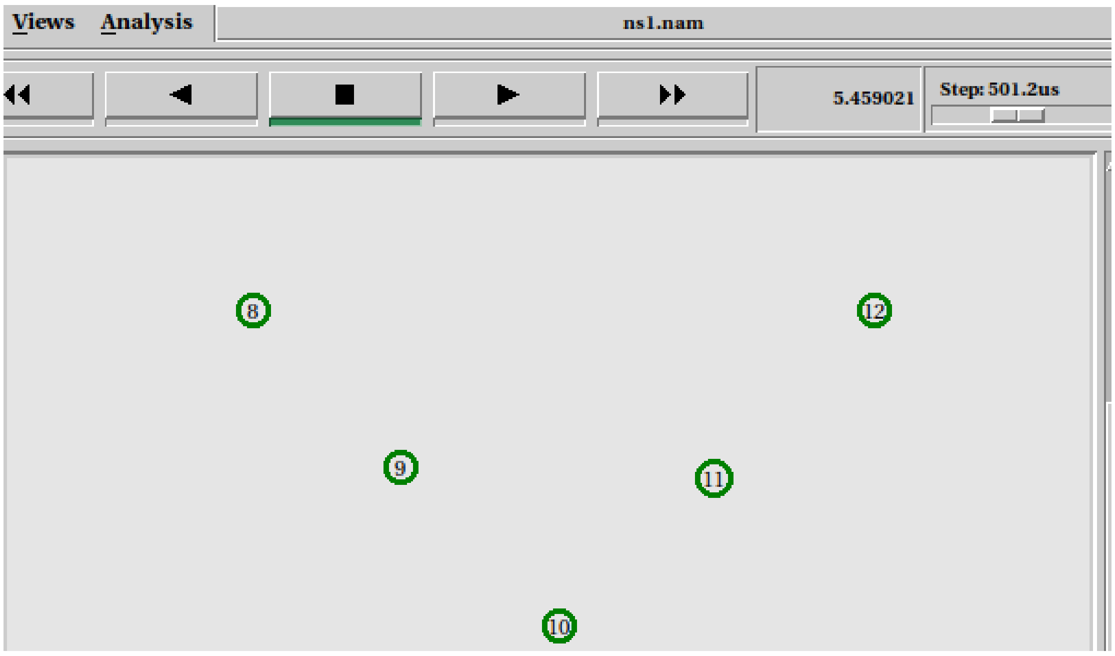
I guess the reason why packet loss occurs so quickly is that I set the packet too large and the send interval too small when the network parameters are set, the network load is too large, and the packet loss occurs very quickly.

5. The transmission ends

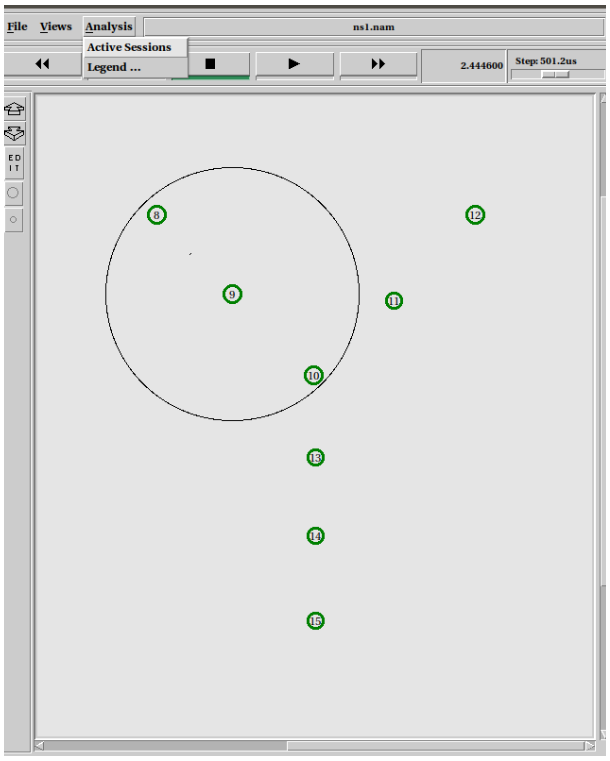
For the SMAC protocol: the transmission ends around 31.2s:



For the MAC protocol of 802.11: The transmission ends around 5.45s:

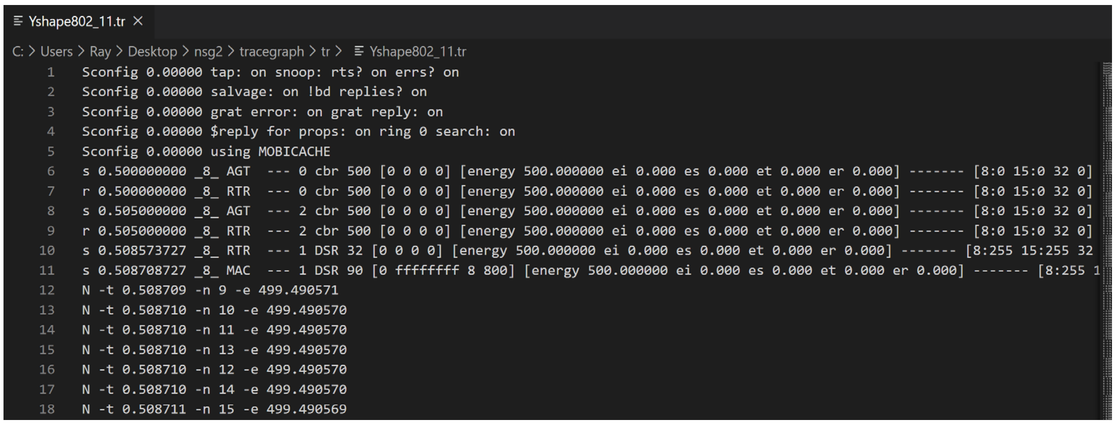


It can be seen that the latency of the SMAC protocol is higher than 802.11, which is consistent with the theoretical results.



VI, trace file analysis

The resulting trace file format is roughly as shown in figure:


Yshape802_11.tr
C: > Users > Ray > Desktop > nsg2 > tracegraph > tr > Yshape802_11.tr

```
1    Sconfig 0.00000 tap: on snoop: rts? on errs? on
2    Sconfig 0.00000 salvage: on !bd replies? on
3    Sconfig 0.00000 grat error: on grat reply: on
4    Sconfig 0.00000 $reply for props: on ring 0 search: on
5    Sconfig 0.00000 using MOBICACHE
6    s 0.500000000 _8_ AGT  --- 0 cbr 500 [0 0 0 0] [energy 500.000000 ei 0.000 es 0.000 et 0.000 er 0.000] ------- [8:0 15:0 32 0]
7    r 0.500000000 _8_ RTR  --- 0 cbr 500 [0 0 0 0] [energy 500.000000 ei 0.000 es 0.000 et 0.000 er 0.000] ------- [8:0 15:0 32 0]
8    s 0.505000000 _8_ AGT  --- 2 cbr 500 [0 0 0 0] [energy 500.000000 ei 0.000 es 0.000 et 0.000 er 0.000] ------- [8:0 15:0 32 0]
9    r 0.505000000 _8_ RTR  --- 2 cbr 500 [0 0 0 0] [energy 500.000000 ei 0.000 es 0.000 et 0.000 er 0.000] ------- [8:0 15:0 32 0]
10   s 0.508573727 _8_ RTR  --- 1 DSR 32 [0 0 0 0] [energy 500.000000 ei 0.000 es 0.000 et 0.000 er 0.000] ------- [8:255 15:255 32
11   s 0.508708727 _8_ MAC  --- 1 DSR 90 [0 ffffffff 8 800] [energy 500.000000 ei 0.000 es 0.000 et 0.000 er 0.000] ------- [8:255 1
12   N -t 0.508709 -n 9 -e 499.490571
13   N -t 0.508710 -n 10 -e 499.490570
14   N -t 0.508710 -n 11 -e 499.490570
15   N -t 0.508710 -n 13 -e 499.490570
16   N -t 0.508710 -n 12 -e 499.490570
17   N -t 0.508710 -n 14 -e 499.490570
18   N -t 0.508711 -n 15 -e 499.490569
```

Where the first five industries should be the default file header, starting from the sixth line, the first letter represents the type of event that occurred.

S stands for send that sends events

R stands for resume, that is, accepts the event

D stands for discard, i.e. drop event

F stands for forwarding events

The second column is the time at which the represented event occurred

The third column is the node number

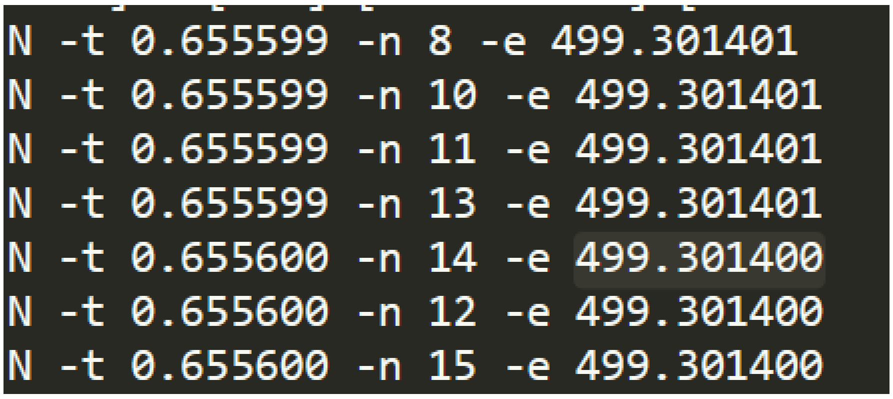
The fourth column is the type of trace

The AGT should be the agent trace representing the trace of the agent proxy in ns2

RTR is Router trace, which stands for trace to a route
Mac is the trace of the MAC layer
Column V ---- has no practical significance
The sixth column is the ID of the group
The seventh column is the type of grouping
The eighth column is the group size unit that is byte
In the ninth column square brackets [] are the details of the MAC layer
In the tenth column square brackets [] are the details of the energy model, taking the information in the sixth row as an example 500 ei indicates the intrinsic energy 500J, 0 es is sleeping energy, et and so on for transit energy, er for reapiving energy
The square brackets of the last column take the sixth line as an example, "8:0" is the "node number: port number" sent in groups, "15" :0" is the "node number:port number" for the purpose. 32 is the value of Time to Live, and each TTL value is minus 1 to 0 The default packet is lost and retransmission is enabled. The last number zero is the number of hops from the source node to the destination node.

● Energy and delay analysis

Take the energy field in the trace file obtained by the SMAC protocol as an example:

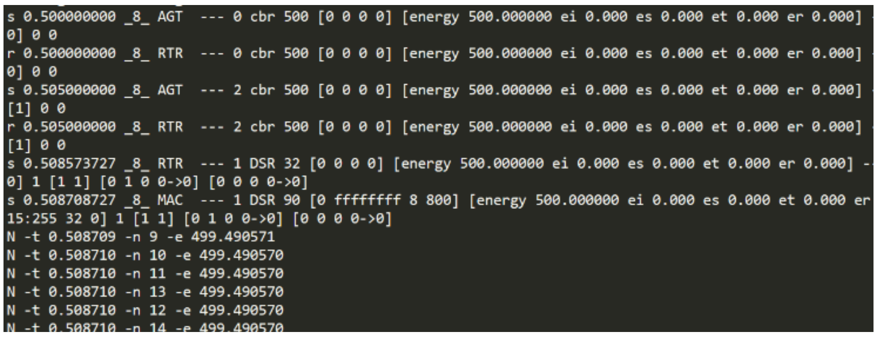


```
N -t 0.655599 -n 8 -e 499.301401
N -t 0.655599 -n 10 -e 499.301401
N -t 0.655599 -n 11 -e 499.301401
N -t 0.655599 -n 13 -e 499.301401
N -t 0.655600 -n 14 -e 499.301400
N -t 0.655600 -n 12 -e 499.301400
N -t 0.655600 -n 15 -e 499.301400
```

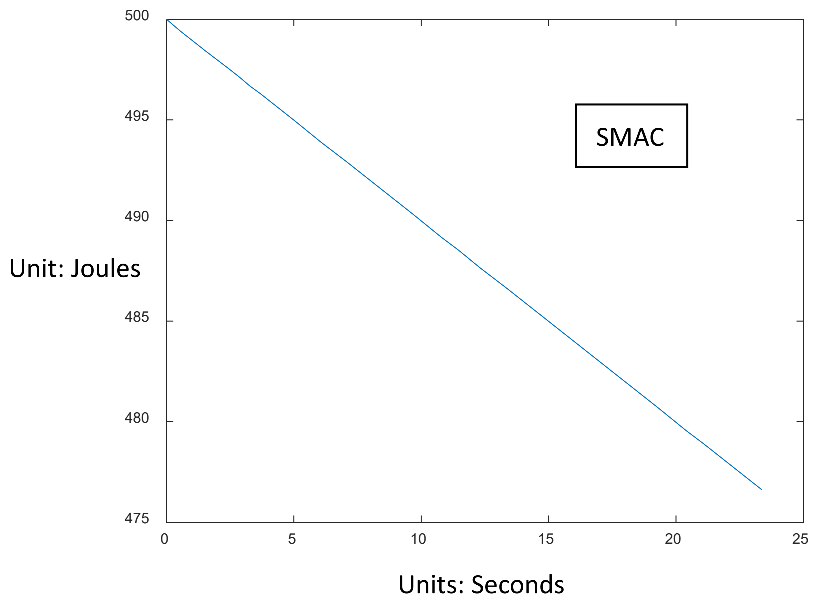
\

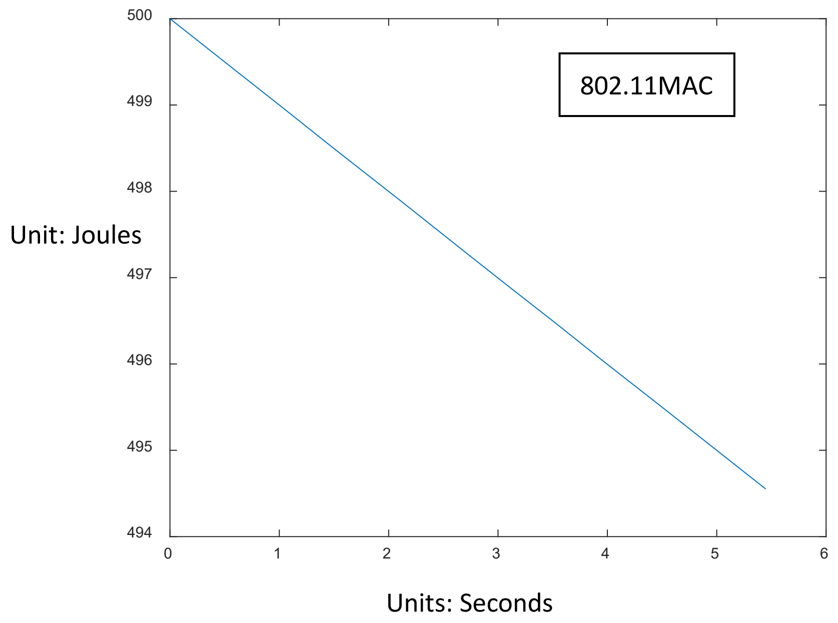
Take Y shape smac For example, N -t 0.655599 represents the simulation time 0.655699s, -n 8, for the first transmit energy trace report Represents node 8 and -e 499.3 represents the remaining energy of the system

Since the energy model model I added was developed by ns2 contributors, its trace output format is different from the original trace file output, which results in the use of matlab or When the awk script extracts data, the data format is not uniform, and the two formats are mixed, which makes it difficult for data analysis. The energy information in the SMAC protocol and the 802_11 protocol is plotted with MATLAB as follows
Manual sampling is employed(one per second)Way to draw energy Figure, sample data see.mcode



```
s 0.500000000 _8_ AGT  --- 0 cbr 500 [0 0 0 0] [energy 500.000000 ei 0.000 es 0.000 et 0.000 er 0.000]
0] 0 0
r 0.500000000 _8_ RTR  --- 0 cbr 500 [0 0 0 0] [energy 500.000000 ei 0.000 es 0.000 et 0.000 er 0.000]
0] 0 0
s 0.505000000 _8_ AGT  --- 2 cbr 500 [0 0 0 0] [energy 500.000000 ei 0.000 es 0.000 et 0.000 er 0.000]
[1] 0 0
r 0.505000000 _8_ RTR  --- 2 cbr 500 [0 0 0 0] [energy 500.000000 ei 0.000 es 0.000 et 0.000 er 0.000]
[1] 0 0
s 0.508573727 _8_ RTR  --- 1 DSR 32 [0 0 0 0] [energy 500.000000 ei 0.000 es 0.000 et 0.000 er 0.000] -
0] 1 [1 1] [0 1 0 0->0] [0 0 0 0->0]
s 0.508708727 _8_ MAC  --- 1 DSR 90 [0 ffffffff 8 800] [energy 500.000000 ei 0.000 es 0.000 et 0.000 er
15:255 32 0] 1 [1 1] [0 1 0 0->0] [0 0 0 0->0]
N -t 0.508709 -n 9 -e 499.490571
N -t 0.508710 -n 10 -e 499.490570
N -t 0.508710 -n 11 -e 499.490570
N -t 0.508710 -n 13 -e 499.490570
N -t 0.508710 -n 12 -e 499.490570
N -t 0.508710 -n 14 -e 499.490570
```





It can be seen from this diagram that under this topology, the end-to-end delay of the SMAC protocol is about 24s, and the end-to-end delay of 802.11 is about 5.5 seconds.

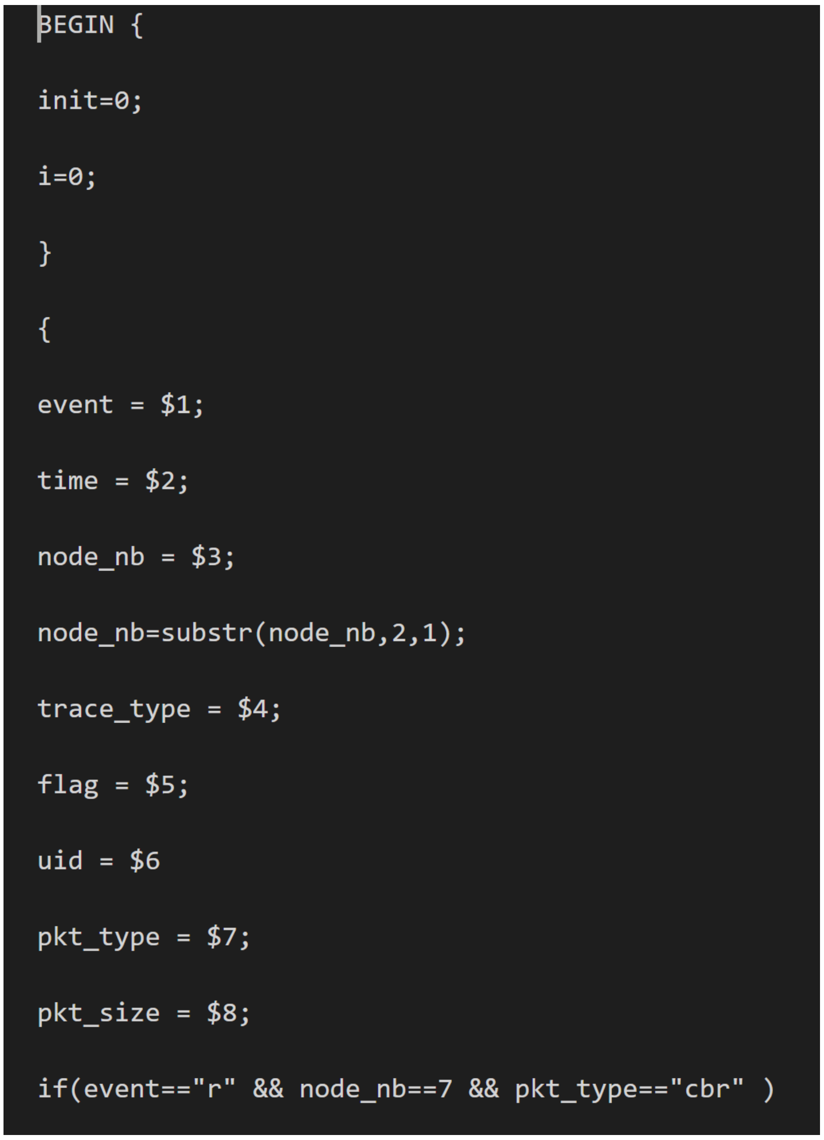
For the consumption of system energy, the energy consumed using the SMAC protocol is about 24J, and the energy consumed using 802.11 is about 5.5J

The energy sent and received by the setting is set to 1J, resulting in a curve that is linear and not quite realistic.

● Throughput and packet loss rate

In the analysis of throughput and packet loss rate, due to its lack of large-scale formatting data processing capabilities, coupled with the confusion of trace script format after the introduction of energy model, it is also confused Modifying the script example of awk on CSDN to run the trace file will not even get the output....

Take this awk script as an example:



```
BEGIN {

init=0;

i=0;

}

{

event = $1;

time = $2;

node_nb = $3;

node_nb=substr(node_nb,2,1);

trace_type = $4;

flag = $5;

uid = $6

pkt_type = $7;

pkt_size = $8;

if(event=="r" && node_nb==7 && pkt_type=="cbr" )
```

Whether I add a new identifier to the energy model or try to skip the energy model information field, I can't get the correct output....

But at least according to the NAM simulation, I guess the simulated packet loss rate is very high. Both the SMAC and 802.11 protocols have a continuous loss of packets at the source node less than 1 second after the simulation starts, and that's why Due to the lack of actual network design experience, I always feel that the transmission interval setting of the CBR traffic generator in the initial network parameter setting is too low, and the transmission interval is too short. Another possible reason is that since ns2 emulation uses the UDP connectionless transport layer protocol, which is itself an unreliable connection, the packet loss rate is naturally better than that used The TCP protocol network is much higher

VII. Summary

This experiment gave me a deeper understanding of the characteristics of the SMAC protocol at the theoretical level, and the experimental results are basically the same as the theoretical results. At the same time, it is also an introduction to the Tcl language and network emulator, and further familiarizes himself with the use of the Linux operating system.

The shortcomings and points that can be continued to improve this experiment are:

1、 After the LEACH protocol is successfully compiled, due to the unfamiliarity with the ns2 software structure, an error occurred when modifying the library, resulting in the inability to call the REACH protocol in the Tcl script. It is possible to reinstall ns2 later, but the newer generation of network emulation software ns3 should be a friendlier option.

2、 Although I introduced the Energy model in the script to be able to analyze the energy, due to insufficient experience in data processing, it was not uniform to successfully export the format using the awk script trace individual data in the file to make a more accurate chart. In addition, the energy parameter settings of the Energy model may not be reasonable, and you can increase your experience by doing more network-related battles in the future.

3、 The knowledge of computer networks is not systematic enough, and many network parameters and protocols are poorly understood when modifying Tcl routines.

4、 You can also try more network topologies that are not limited to small networks of 8 nodes. A step further can be to try building WSN battles using real embedded devices.

References:

[1] LEACH protocol installation in ns2 (ns-2.35) https://www.nsnam.com/2015/05/leach-protocol-installation-in-ns2-ns.html

[2] Energy model in ns2 http://www.jgyan.com/ns2/energy%20model%20in%20ns2.php https://www.isi.edu/nsnam/ns/doc/node224.html

[3]NS2 trace file analysis
https://blog.csdn.net/xiao_sheng_jun/article/details/79683098

[4] NS2 Tutorial https://blog.csdn.net/xiahouzuoxin/article/details/16959637

[5] 'NS Simulator for beginners'

[6]Tcl Tutorial https://www.yiibai.com/tcl