

# Lab4 Report

楊行 r03942126, 葉金龍 r03922140, 秦建平 r03922119, 嚴俊軒 r02942020, 傅勝余 d03922013

## 1 Target

We are engaged to build an Ad-hoc network with Zigduino. As in the picture, we will use Zigduino placed on fourth floor to build such a network. This network should make the source node able to communicate with destination node by multi-hop.

Source should be able to find the route by himself, which is called routing, and then ping the destination. The benchmark is the round-trip-time (RTT) and success rate of packet transmission.

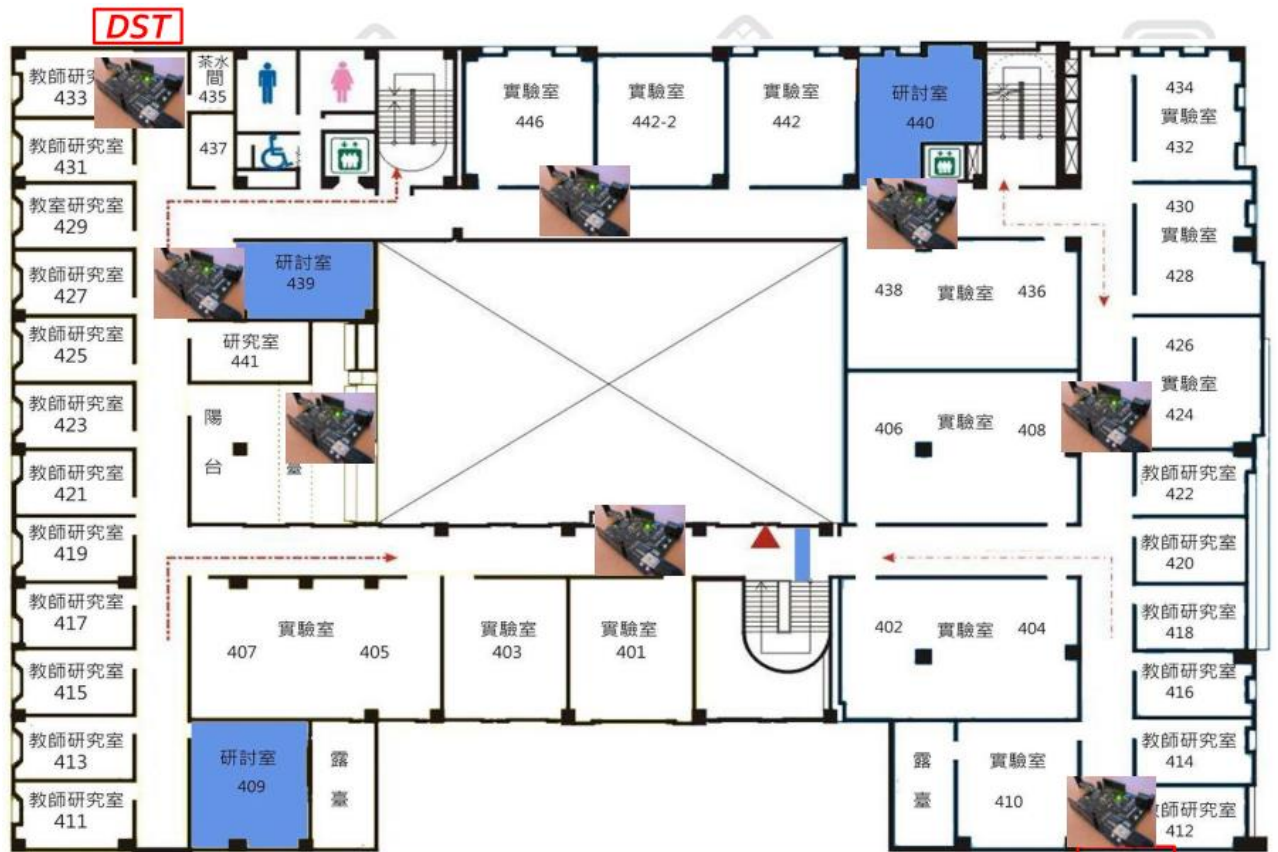


Figure 1 Placement of nodes

## 2. To do

To get to the target, we need two things, robust link and routing.

### 2.1 Robust link

Robust link is the fundamental of routing. Because of weak quality of wireless channel and shared wireless channel, the link is very weak. Packets may corrupt as result of collision and attenuation. Therefore, we need to use some mechanisms to build robust link, including ACK/retry, error detection, CSMA.

### 2.1.1 ACK & retransmission

#### ACK

ACK is used to tell transmitter packet has been received correctly. If not, transmitter should retry some times until receiving ACK packet or timeout.

In this lab, Zigduino has two kinds of ACK, hardware ACK, and software ACK. Hardware ACK is fast, but it doesn't do no error detection and identification which we are not sure. Software can be used to do such work, but has more latency. And the software ACK is defined by protocol that it is 5 bytes long with first byte of header 0x42. We can't do some more things for example, identification which need to put receiver address into ACK packet. So we build our own software ACK with "normal" packet, using one reserved bit 7 of first byte of header.

#### Retransmission

It is very important to retry when no ACK happens. We set retry times as 5. And if packets need ACK, the bit 5 of first byte will be set, which is as in figure 2.

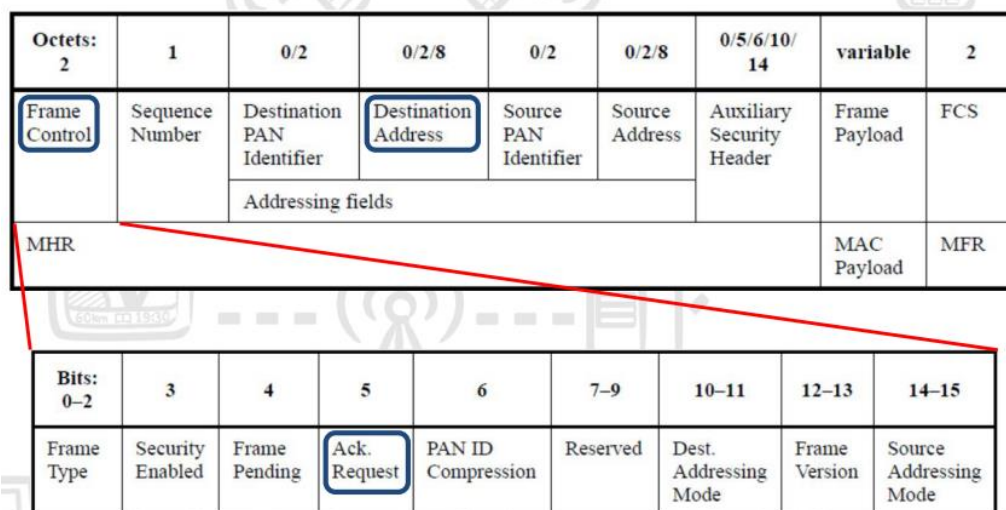


Figure 2 Frame format of packet

### 2.1.2 Error Detection (CRC)

There are many methods to do error detection. The typical one is cyclic redundancy check (CRC)<sup>[1]</sup>. This method is very robust that almost impossible that it makes wrong detection. We realize this function.

### 2.1.3 CSMA/CA

To get efficient transmission, we also need to do some schedule, for example CSMA/CA. It means that "listen before transmit" and random backoff before transmission. We also realize it but we don't see the better performance. The problem here is synchronization. We don't have synchronization. This maybe one reason we don't get better performance when we use CSMA/CA.

## 2.2 Routing

1	2	3	4	5	6	7
RREQ/ RREP	Hop	Route seq	SRC_ID		DST_ID	

Table 1: Routing packet format

1	2	3	4	5	6
RREQ/ RREP	Hop	SRC_ID		DST_ID	

Table 1: Ping packet format

Unit: Byte

RREQ: request of routing

RREP: reply of routing

Hop: number of hops frame go through

Route seq: the ID of broadcast packet for routing

### 2.2.1 Algorithm

Routing part includes two parts, searching the DST and replying, we call them down link (DL) and up link (UL).

We broadcast in DL, and unicast in UL. We use RREQ and RREP to tag the two processes. And each node will record the next hop of the path. The problem is it can not rebuild a path when some node is gone.

#### Routing procedure:

STEP 1: SRC broadcast routing packet with RREQ

STEP 2: node who receives RREQ, record the previous hop, i.e. SRC.

Then he checks whether the des\_addr in header is himself.

If so, rebuild a packet with RREP. Then unicast to previous hop using the record. Jump to ping mode.

If not, broadcast with src\_addr in header changed to himself.

STEP 3: node who receives RREP record the previous hop until this packet is transmitted to SRC.

Jump to ping mode to transmit data.

STEP 4: SRC receives the routing packet with RREP, he just knows the routing is finished. Then jump to ping mode.

STEP 5: SRC sends data packet with PING, format is as table 2.

STEP 6: node who receives data packet will add the hop with 1 and forward it to next hop by record

STEP 7: DST receive the ping packet then reply.

## 2.2.2 Hop

The hop in each packet means “how long” does the packet exists in the network, added by 1 after one hop. Because of limited scale of network, we define a HOP\_MAX as value of 4. It means that packets can only be transmitted for four hops.

This mechanism reduces the flooding of packets made by broadcast and loop.

## 3 Demo

Setting:

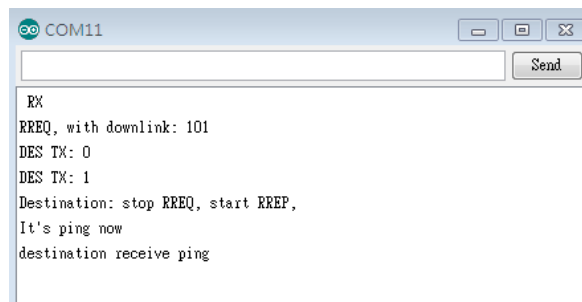
SRC\_ID: 0x0101

DST\_ID: 0x0103

Fig3 shows source node broadcast first. Until it receives reply from destination, it shows “Route established.” 103 means the previous node is 0103. And the other 103 means the DST is 0103.

Fig4 shows that destination was found and the previous node is 0101. DES TX:0 means it replies first time. And after two times try, show that “Destination: stop RREQ, start RREP”.

```
Broadcast
Broadcast
Broadcast
Broadcast
Broadcast
RX:
Route established
103
Rout END
103
Ping num 0
Ping fail
Ping num 1
Ping fail
Ping num 2
Ping success: 34
Ping num 3
Ping fail
Ping num 4
Ping fail
```



```
COM11
RX
RREQ, with downlink: 101
DES TX: 0
DES TX: 1
Destination: stop RREQ, start RREP,
It's ping now
destination receive ping
```

Figure 4 output of destination node

Figure 3 output of source node

## 4 Problems we have met

### ACK

We first use some time to judge the correctness of hardware ACK. Then under the help of TA and we check many times, we find hardware ACK, doesn't work well, because it doesn't do the identification of the received packet and just ACK autonomously. Therefore we decide to use software ACK.

At first, we thought the defined ACK packet was flexible that we can add payload we want. We found it didn't work. Then we formed the ACK packet as “normal” packet, using the reserved bit 7 of header as a tag, which made header “0xC1”. Then use pkt\_Tx() function to transmit it.

## Serial Monitor

We wasted long time here. Because of limited buffer and bitrate of monitor, Zigduino always corrupt when we print something.

## Few backward packets

When we do the ping, we found much less backward packets. It is because of too often forward packets. Backward packets have less access of channel. This problem was thought to be solved by CSMA/CA. But CSMA/CA doesn't work well here as we talked in previous section.

## Team work

We spend much time and energy on this lab not only because of the hardness of the lab, but more of our poor team work. We are separated. Not enough discussion at first. Then one's work is not continued well. And the code management is poor. Above all, we didn't use the goodness of every member. We need to do many changes in final project.

## Reference:

[1] [http://en.wikipedia.org/wiki/Cyclic\\_redundancy\\_check](http://en.wikipedia.org/wiki/Cyclic_redundancy_check)

[2] Source code: see the code file, src for source node, des for other nodes.