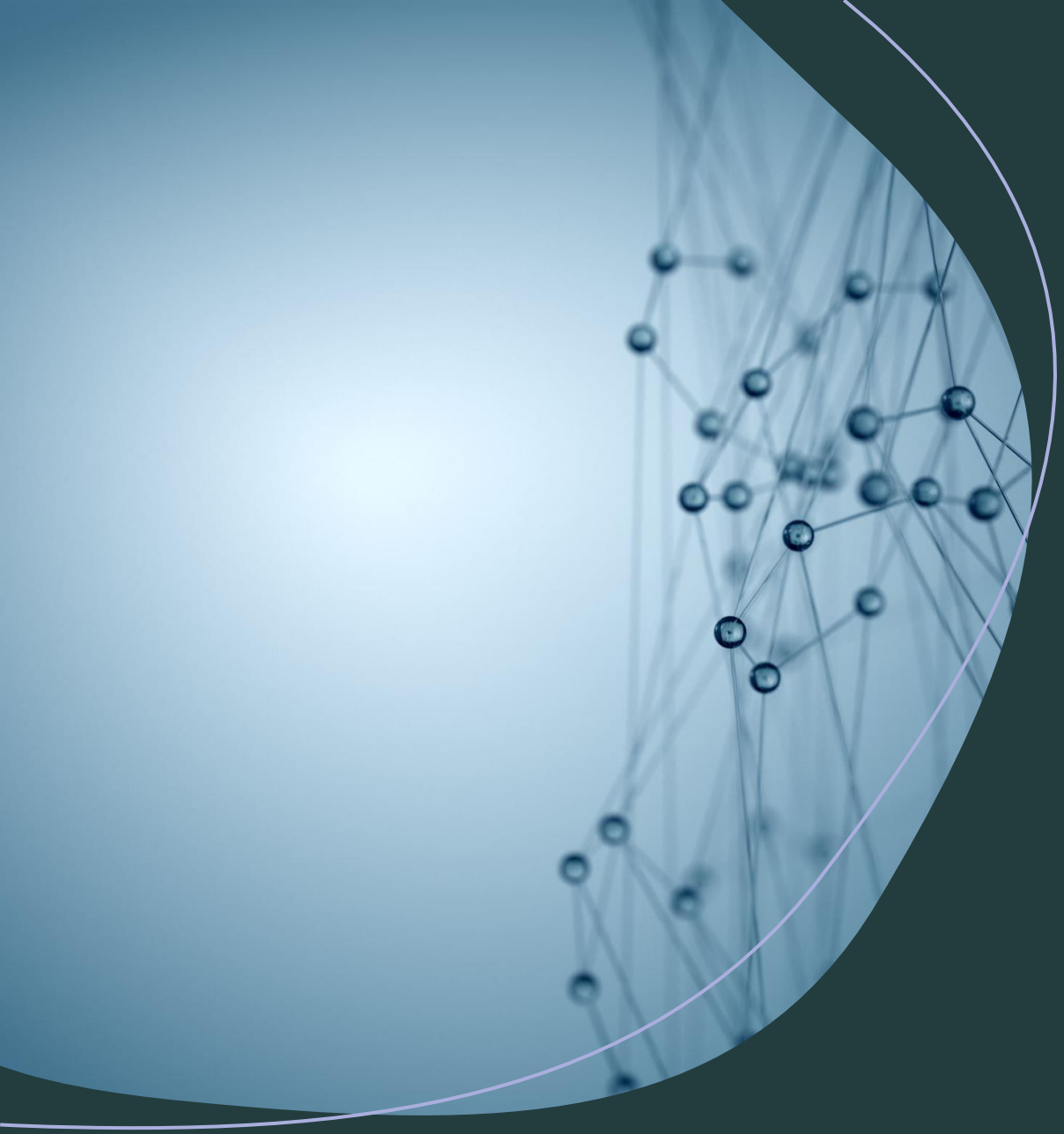
An enhanced design for packet integrity attack detection on Untrusted Network-on-Chip (NoC)

Hardware Trojan Detection & Localization

Accessor : Sri Parameswaran

Supervisor : Hui Guo

Student: Meisi Li



Background

Introduction of Network-on Chip(NoC)

- Network-based communication system
- Improves the scalability and complexity
- Excellent parallelism
- Hidden trouble: hardware Trojan

Problem Statement

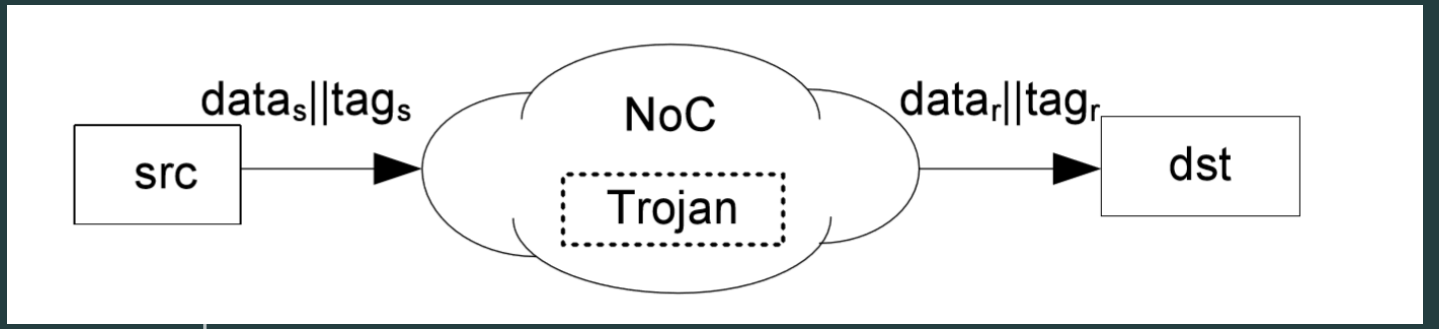
- The task was to design an improved trojan detection method in NoC, based on the existing detection and localization ideas.

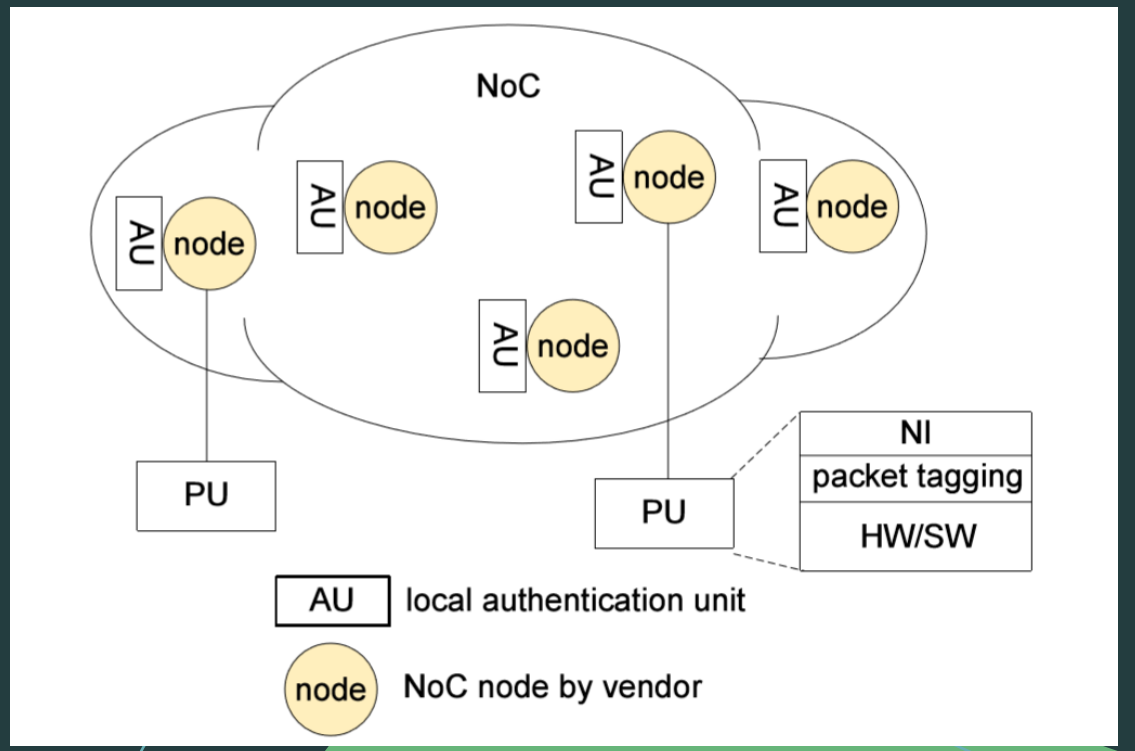
Related works

- Progressive Packet Authentication – Detection
- Energy Efficient Trojan Detection(EETD) – Localization

Related Work 1: Authentication

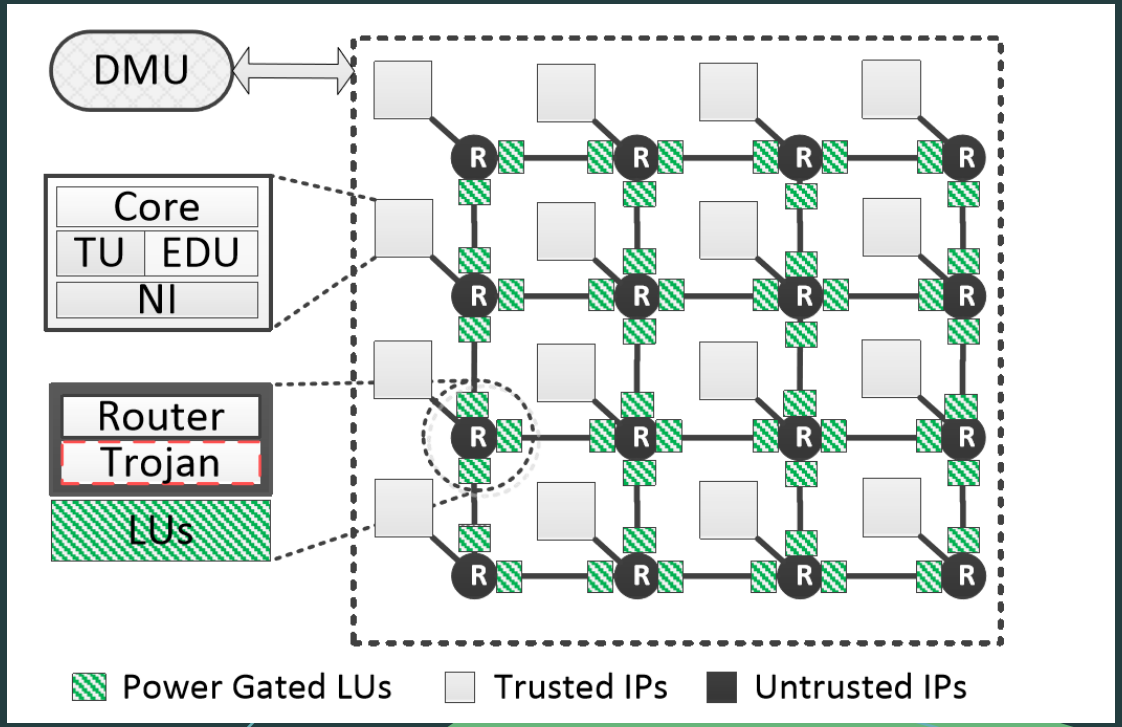
- Traditional
  - Tag the packet data at the source node
  - Transferred over the network to the destination
  - The packet data is authenticated against the tag
- Progressive packet authentication
  - Each node has an authentication unit(AU)
  - Local key in every node





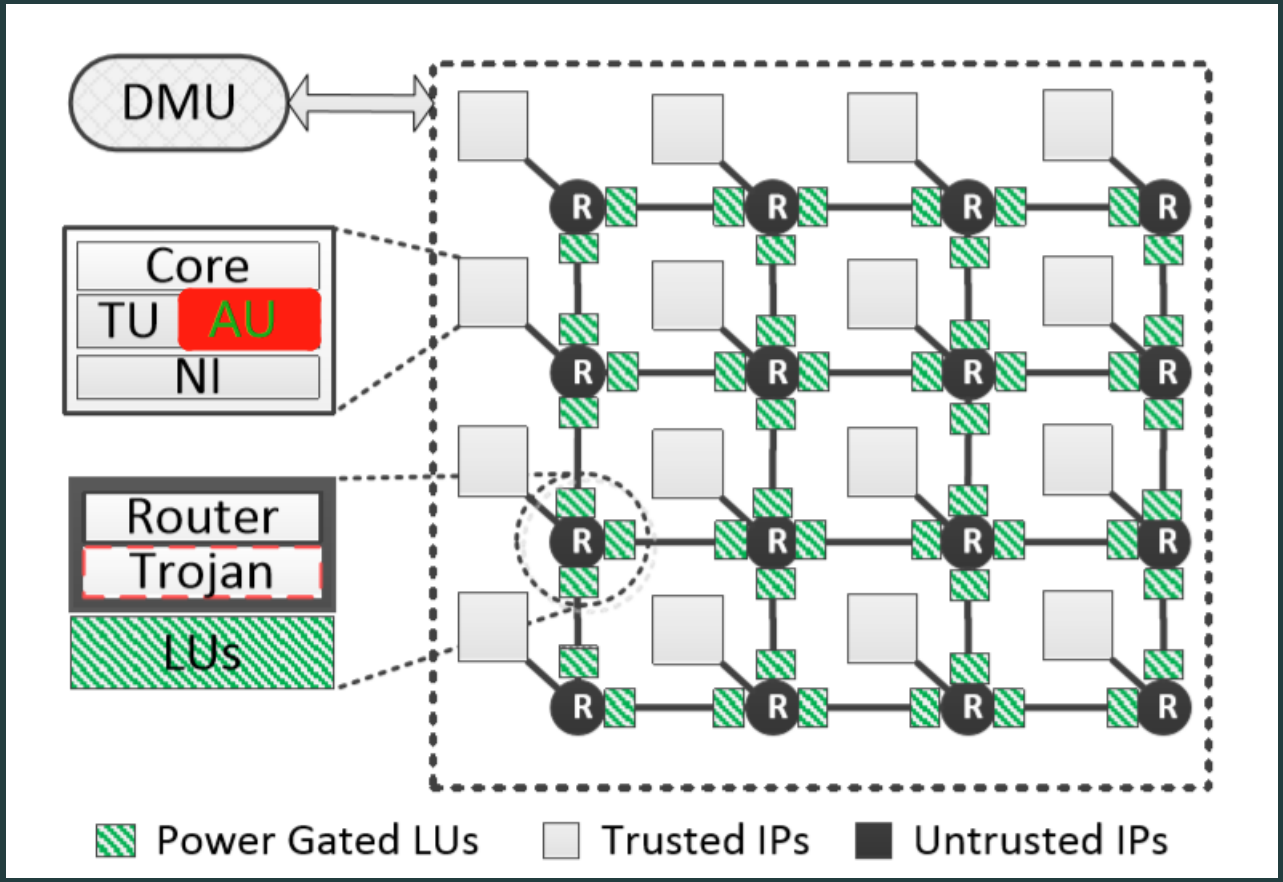
Related work 2: EETD – Energy Efficient Trojan Detection

- Uses selective activation/deactivation of detection units to reduce energy overheads.
- Worm based search algorithm
- Two types of detection units
  - E2E Trojan Detection Units: EDU
  - Localization units: LU



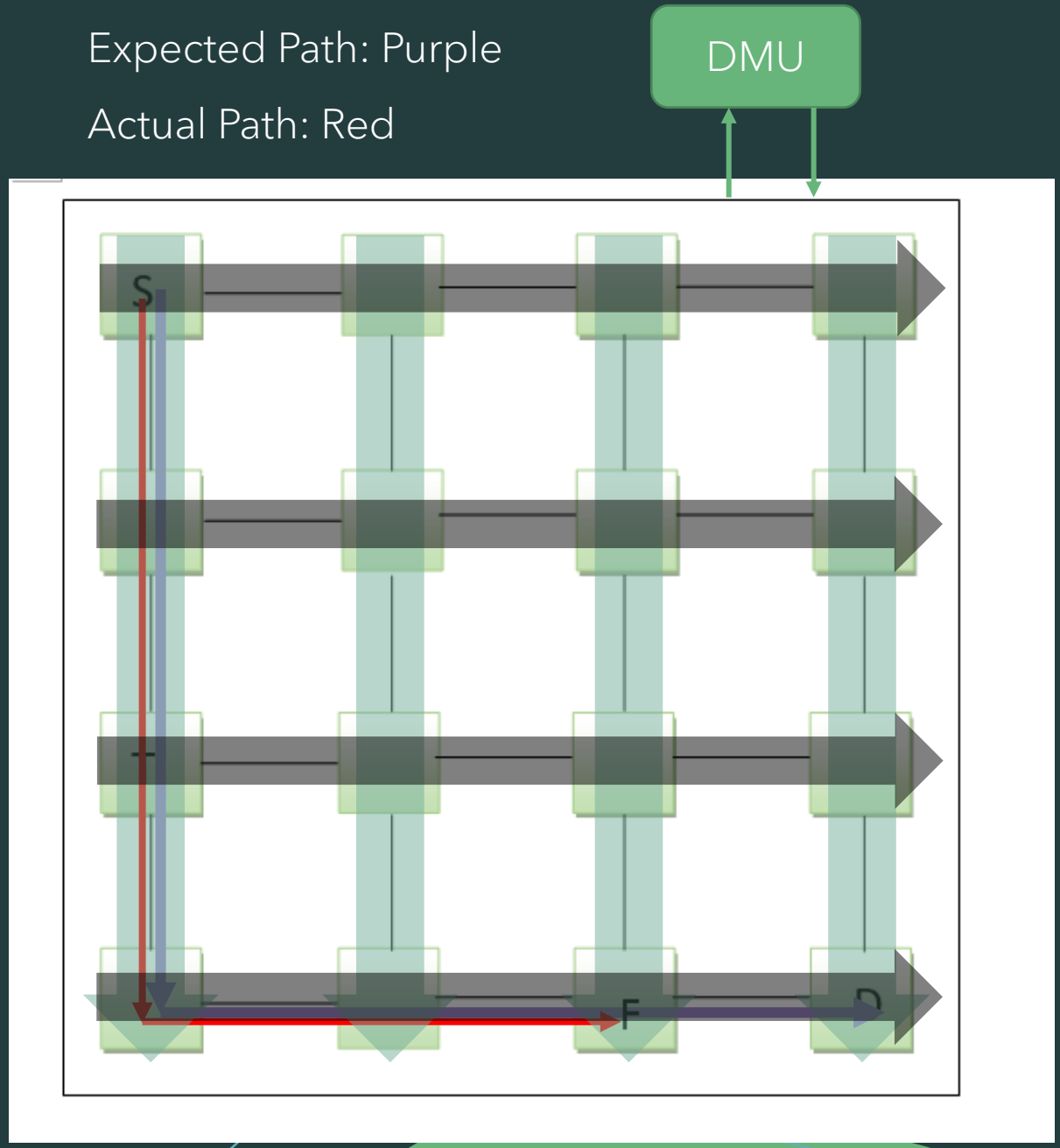
Solution

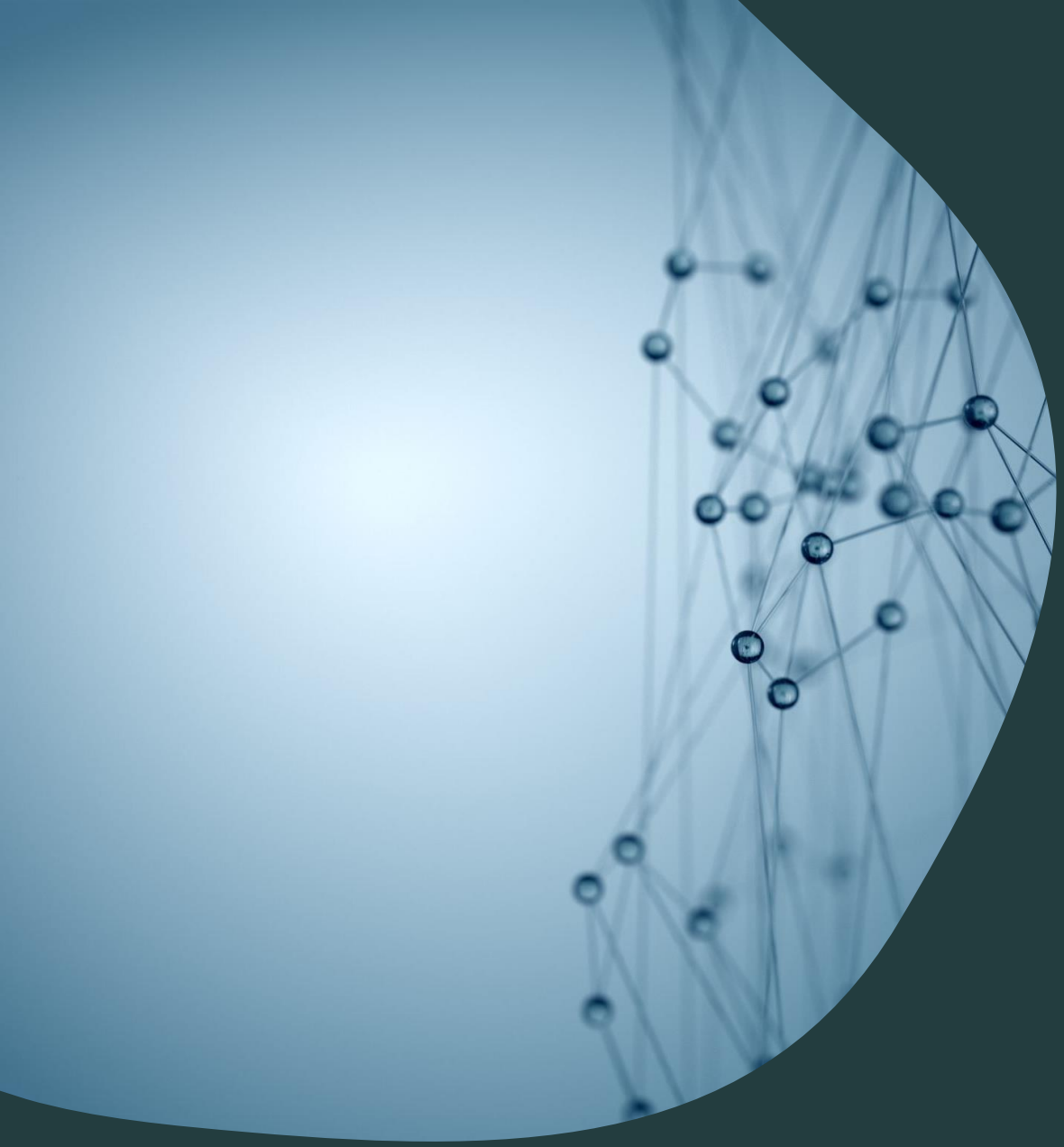
- Enhanced Design
- Replace EDU with AU



Example Network – Localization Unit (LU)

- S: source node
- D: destination node
- T: Trojan
- F: detects the tampered packet
- Search all rows and columns at a time
- The trojan locates in the intersection





Components

Architecture

- Packet format
- Network adapters
  - Master
  - Slave
  - Route calculation
  - Interface
- Routers
  - Routing algorithm
- Detection Unit
- Tag table
- Key table
- Python drawing

Packet Format

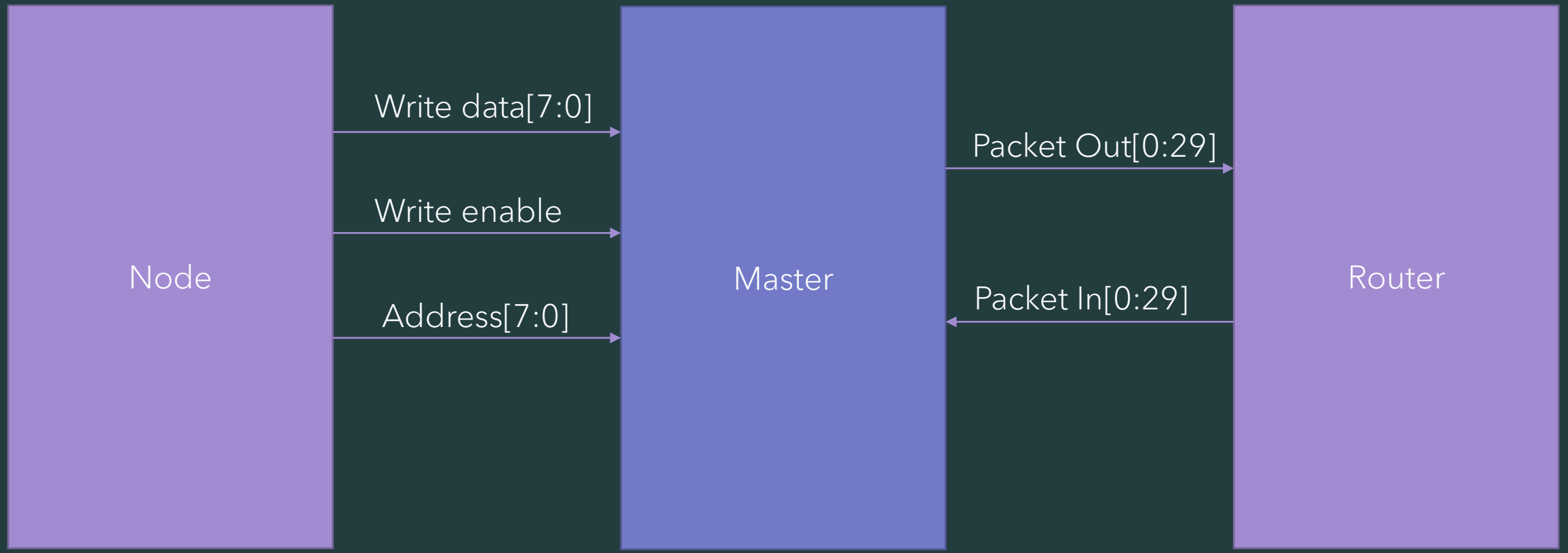
- Packet type: write packet
- Length: 49 bits
- Router Counting: bit-48 to bit-43
- Address in memory: bit-35 to bit-28

48 - 43	42 - 40	39-36	35 - 28	27-0
x/y	flags	Target ID	address	data

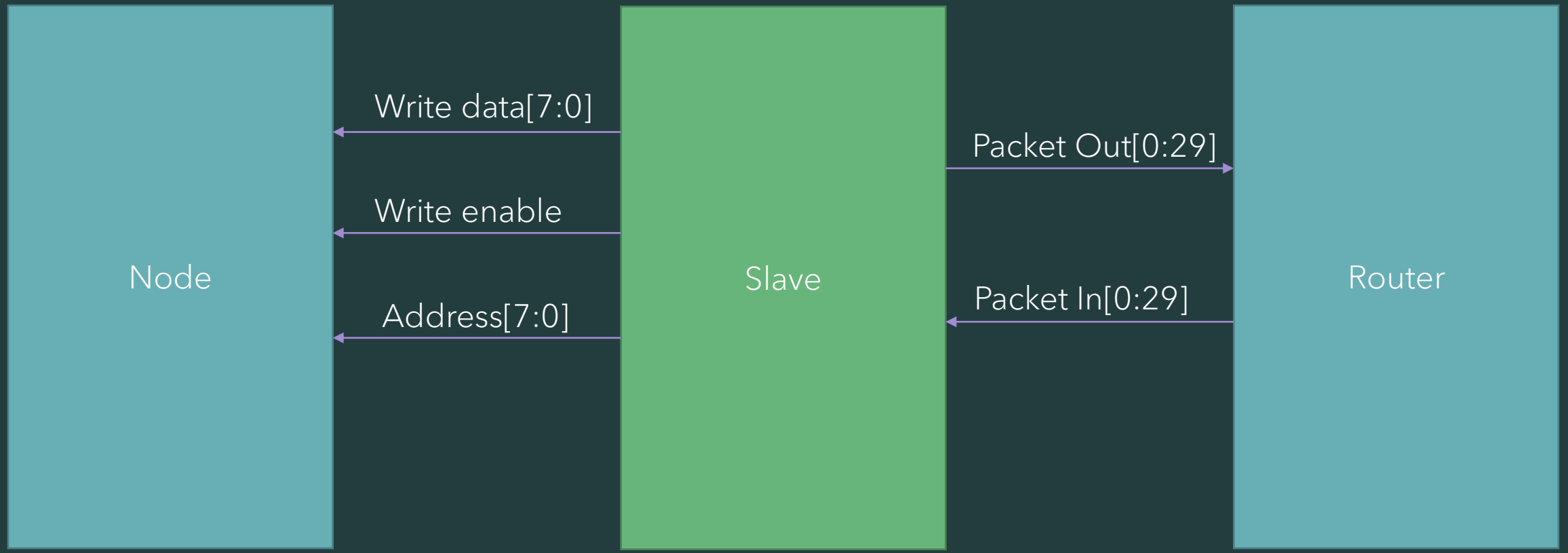
- Packet: 1010 to 1100, with data x"0000111" in addr x"00"

110001	010	1100	0000000	x"0000111"

Master


Node
Write data[7:0]
Write enable
Address[7:0]
Master
Packet Out[0:29]
Packet In[0:29]
Router

Slave


Node
Write data[7:0]
Write enable
Address[7:0]
Slave
Packet Out[0:29]
Packet In[0:29]
Router

48	47-46	45	44-43
y_sign	y_count	x_sign	x_count

Route Calculation

- Master determines the router counting
- y_sign, x_sign: next direction
  - Positive: 0; Negative: 1
- y_count, x_count: the rest moves
- Packet: 1010 to 1100, with data x"0000111" in addr x"00"

48 - 43	42 - 40	39-36	35 - 28	27-0
110001	010	1100	0000000	x"0000111"

28	27-26	25	24-23
y_sign	y_count	x_sign	x_count

Router Algorithm

- Direction: north, east, south, west

```
if  packet_in(22) = '0' and packet_in(21) = '0' and packet_in(20) = '0' then
        direction := 6; -- null
elsif packet_in(27 downto 26) /= "00" then
    if packet_in(28) = '1' then
        direction := 1; -- north
    else
        direction := 3; -- south
    end if;
elsif packet_in(24 downto 23) /= "00" then
    if packet_in(25) = '1' then
        direction := 4; -- west
    else
        direction := 2; -- south
    end if;
else
    direction := 0;     -- local
end if;
```

Interface

- Master write to Slave

```
-- write packet from 1111 to 0000 in addr x"00"
wr <= '1';
targetid <= "0000";
wr_data <= "01110011";
addr <= x"00";
```

- Slave read the write packet

```
if wr = '1' then
    register_array(to_integer(unsigned(addr(7 downto 0)))) <= wr_data(7 downto 0);
end if;
```

Detection Unit

- ThresholdTime:
  - $T = a * b * t_0$
  - a: number of input ports
  - b: number of buffer levels in a node => 1
  - t_0: basic packet latency

```
1   Localization Algorithm:
2   input:  The node location where the first tampered packet
3               is detected, (x_f, y_f)
4           NoC Network
5           NoC Parameters
6   outputs:Trojan Location(x_h, y_h)
7
8   T = calThresholdTime(x_f, y_f)
9   activateLU()
10
11  while waitTime < T and Trojanrow is not found
12          and TrojanCol is not found do
13      y = wormMoveAlongCol()
14      x = wormMoveAlongRow()
15  end while
16
17  (x_h, y_h) = (x, y)
```

Tag table

- sig_flagdata = flag + target_id + data

```
--------------------------------------------
 -- tag table
 -- key : 28, 71, 200, 1, 172, 120, 17, 92, 217, 9, 133, 97, 85, 154, 92, 63
 tag_0 : tag_array(0)     <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 1 + to_integer(unsigned(key_array(0)))), 8));
 tag_1 : tag_array(1)     <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 2 + to_integer(unsigned(key_array(1)))), 8));
 tag_2 : tag_array(2)     <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 3 + to_integer(unsigned(key_array(2)))), 8));
 tag_3 : tag_array(3)     <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 4 + to_integer(unsigned(key_array(3)))), 8));
 tag_4 : tag_array(4)     <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 5 + to_integer(unsigned(key_array(4)))), 8));
 tag_5 : tag_array(5)     <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 6 + to_integer(unsigned(key_array(5)))), 8));
 tag_6 : tag_array(6)     <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 7 + to_integer(unsigned(key_array(6)))), 8));
 tag_7 : tag_array(7)     <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 8 + to_integer(unsigned(key_array(7)))), 8));
 tag_8 : tag_array(8)     <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 9 + to_integer(unsigned(key_array(8)))), 8));
 tag_9 : tag_array(9)     <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 10 + to_integer(unsigned(key_array(9)))), 8));
 tag_10 : tag_array(10)   <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 11 + to_integer(unsigned(key_array(10)))), 8));
 tag_11 : tag_array(11) <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 12 + to_integer(unsigned(key_array(11)))), 8));
 tag_12 : tag_array(12)   <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 13 + to_integer(unsigned(key_array(12)))), 8));
 tag_13 : tag_array(13)   <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 14 + to_integer(unsigned(key_array(13)))), 8));
 tag_14 : tag_array(14)   <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 15 + to_integer(unsigned(key_array(14)))), 8));
 tag_15 : tag_array(15)   <= std_logic_vector(TO_UNSIGNED((to_integer(unsigned(sig_flagdata)) + 16 + to_integer(unsigned(key_array(15)))), 8));
```

Key Table

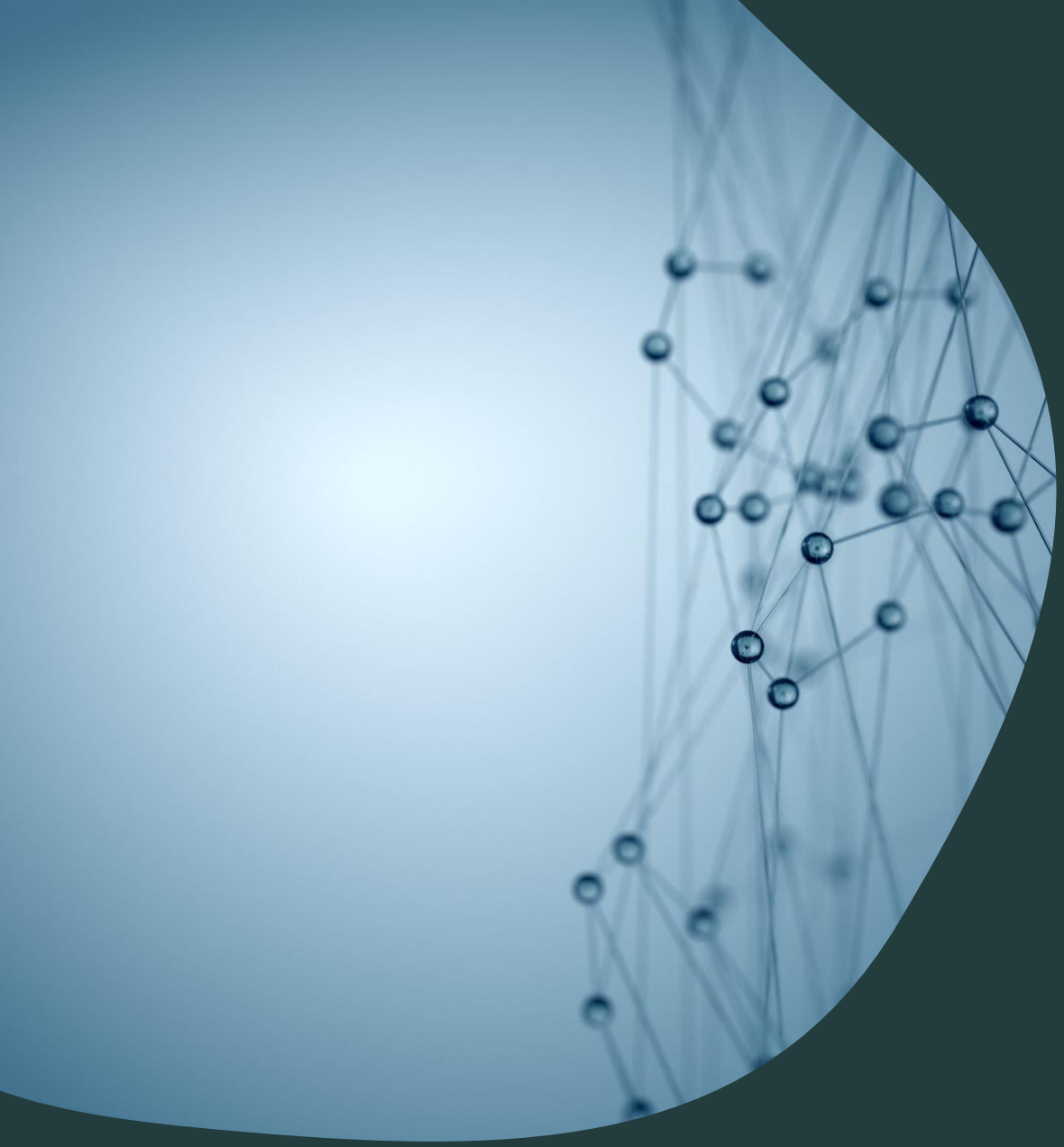
- key :
  - 28, 71, 200, 1,
  - 172, 120, 17, 92,
  - 217, 9, 133, 97,
  - 85, 154, 92, 63

```
key_0 : k0      <= x"1C";
key_1 : k1      <= x"47";
key_2 : k2      <= x"C8";
key_3 : k3      <= x"01";
key_4 : k4      <= x"AC";
key_5 : k5      <= x"78";
key_6 : k6      <= x"11";
key_7 : k7      <= x"5C";
key_8 : k8      <= x"D9";
key_9 : k9      <= x"09";
key_10 : k10    <= x"85";
key_11 : k11    <= x"55";
key_12 : k12    <= x"9A";
key_13 : k13    <= x"5C";
key_14 : k14    <= x"3F";
key_15 : k15    <= x"E7";
```

Python program

- The 'busy' signal from VHDL
  - c_WIDTH: 24
  - File Name: output_result.txt
- Draw the path in the network

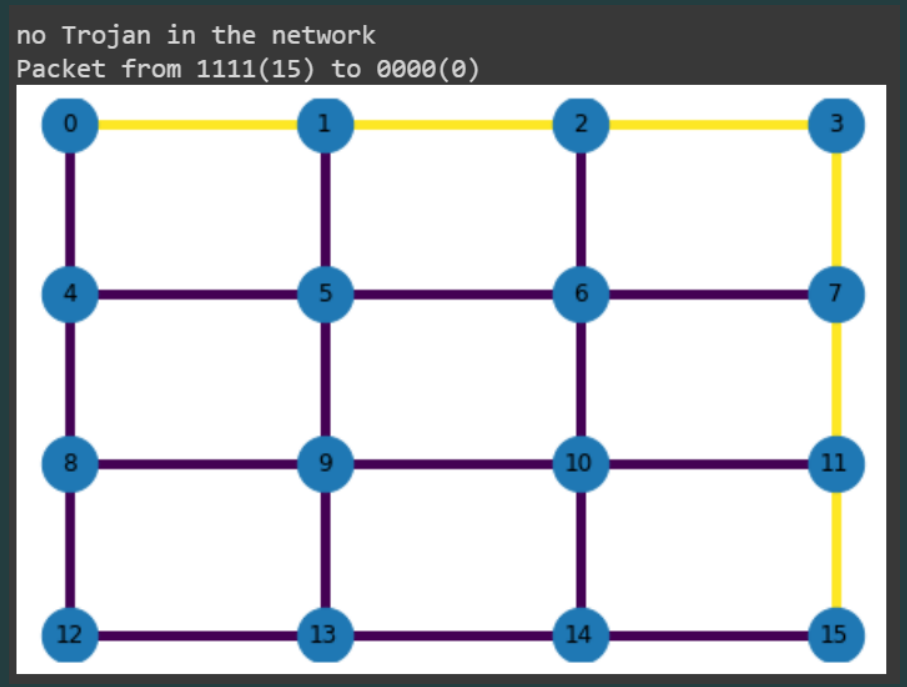
```
N=4
G = nx.grid_2d_graph(N,N)
labels=dict(((i,j),i + (N-1-j)*N) for i, j in G.nodes())
# node id
nx.relabel_nodes(G,labels,False)
inds=labels.keys()
vals=labels.values()
grid_pos=dict(zip(vals,inds)) #Format: {node ID:(i,j)}
plt.figure()
# initial value
df = pd.DataFrame({
        'value':[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]})
# read busy from file
f = open("output_results.txt", 'r')
nums = f.readlines()
nums = [int(i) for i in nums]

# update value
df['value'] = nums

print("Second Packet from 0100 to 1110")
# draw the network
nx.draw(G,pos=grid_pos,with_labels=True,
                edge_color=df['value'],width=5.0,node_size=800)
plt.show()
```



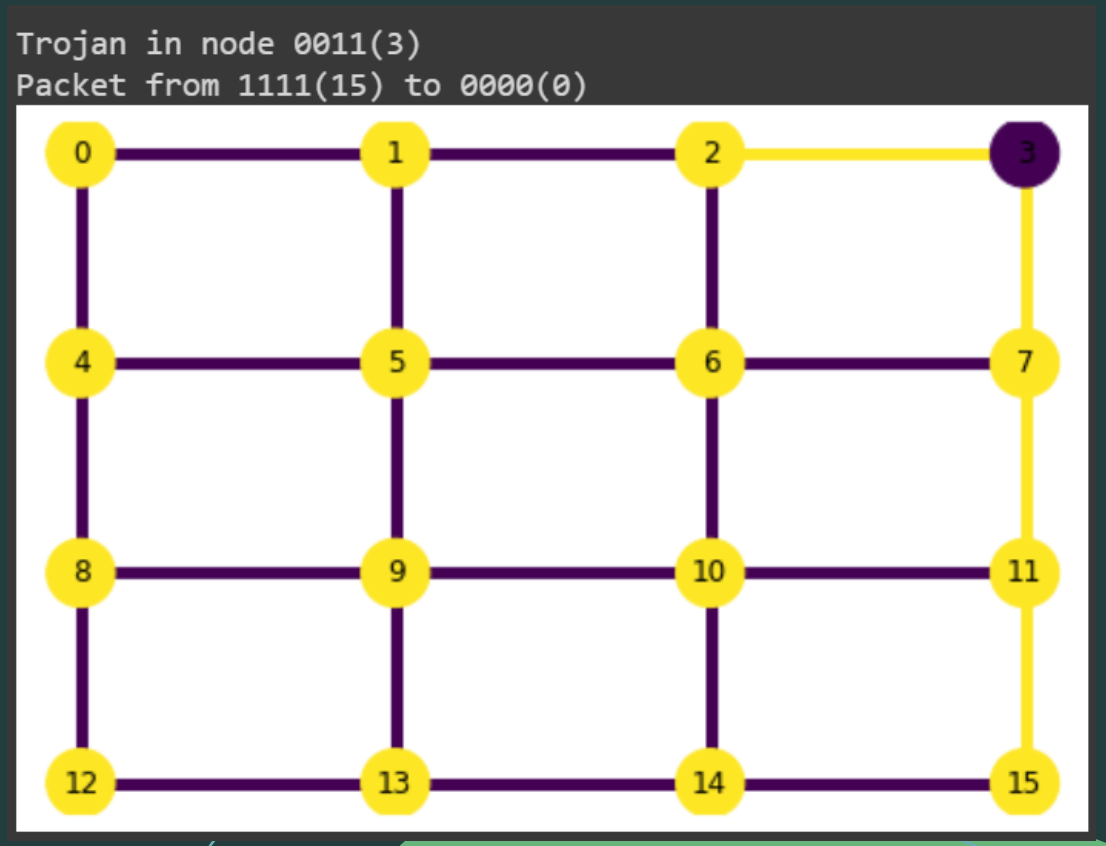
Testing & Evaluation

Testing

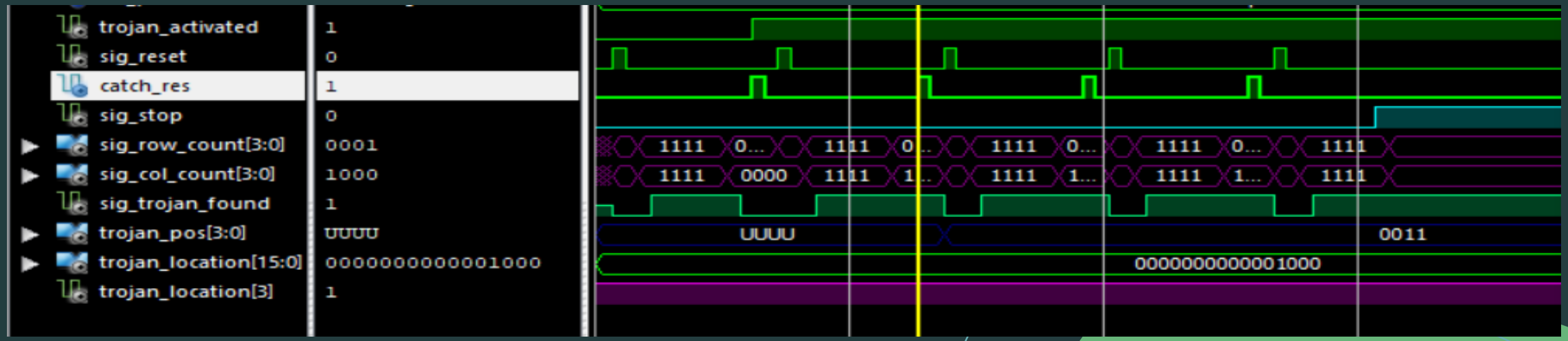
- Test 1: Simple test
  - No trojan in the transfer path



- Trojan in node 3
- Packet is dropped in node 2



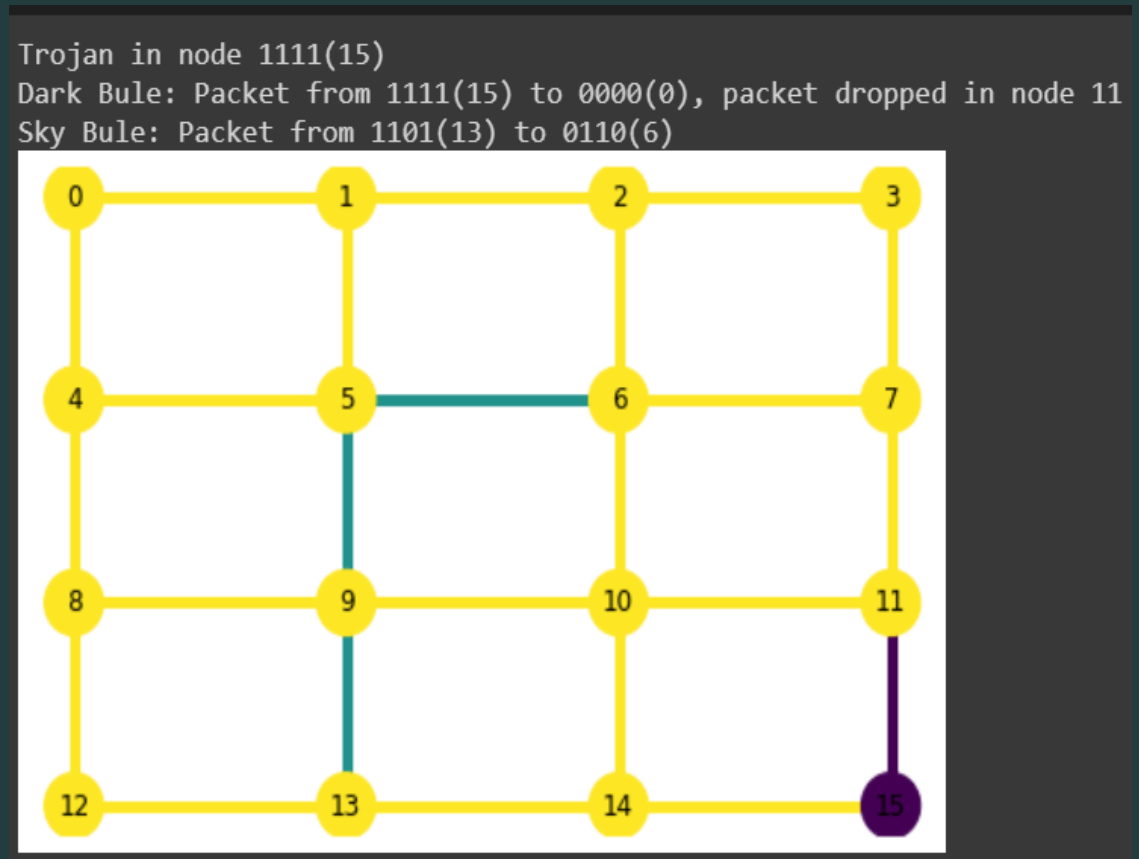
- Job for Localization Unit in test 1
  - trojan_activated: DMU received the packet dropped signal
  - sig_reset: new reset signal to start the detection in the same network
  - catch_res: get the trojan id from the network
  - ‘sig_stop' is 1: the waiting_time(ThresholdTime) is gone
  - sig_row_count/sig_col_count: determines the tampered packets in which row/col
  - ‘sig_trojan_found’: make sure the result is valid
  - 'trojan_pos’: the trojan id from the detection unit
  - 'trojan_location’: original trojan location
    - The position in example is in node 3

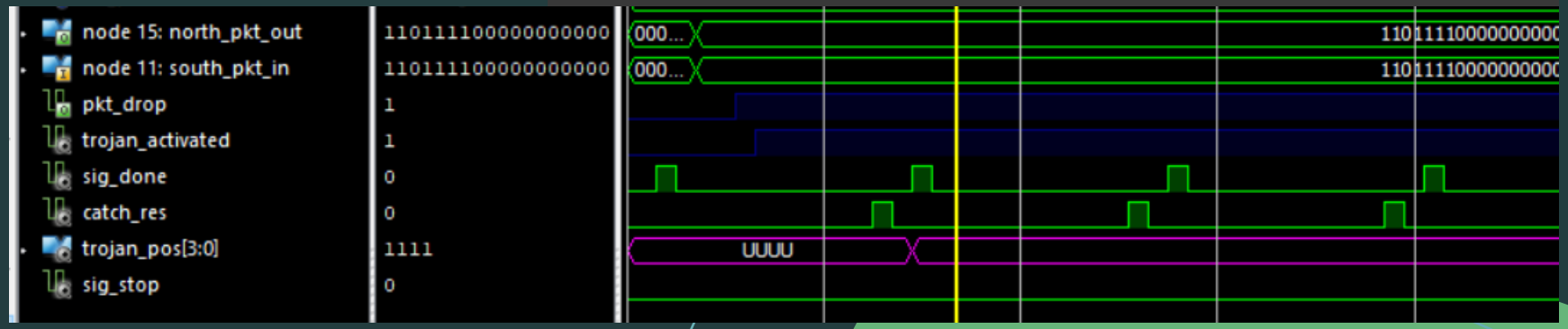


Testing

- Testing 2
  - Packets from varies nodes and some pass through trojan but others not
  - Different trojan position
    - Source node
    - Path node
    - Other nodes

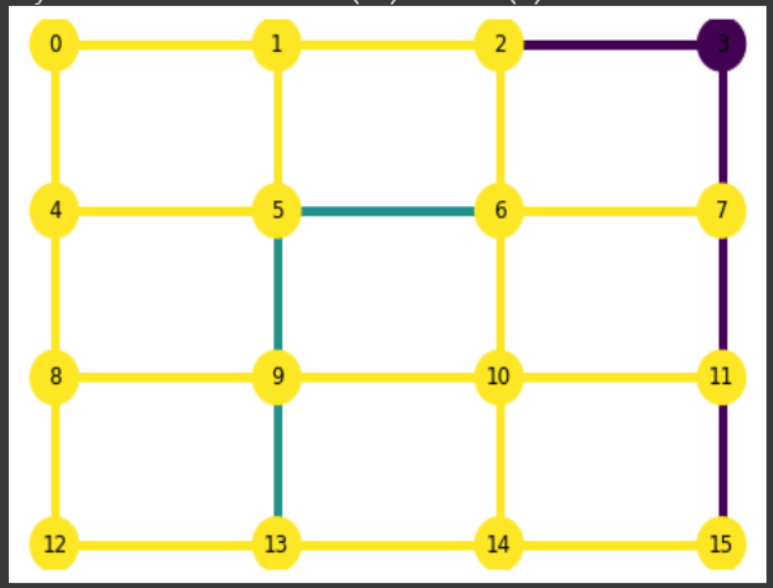
- Source node & Other node





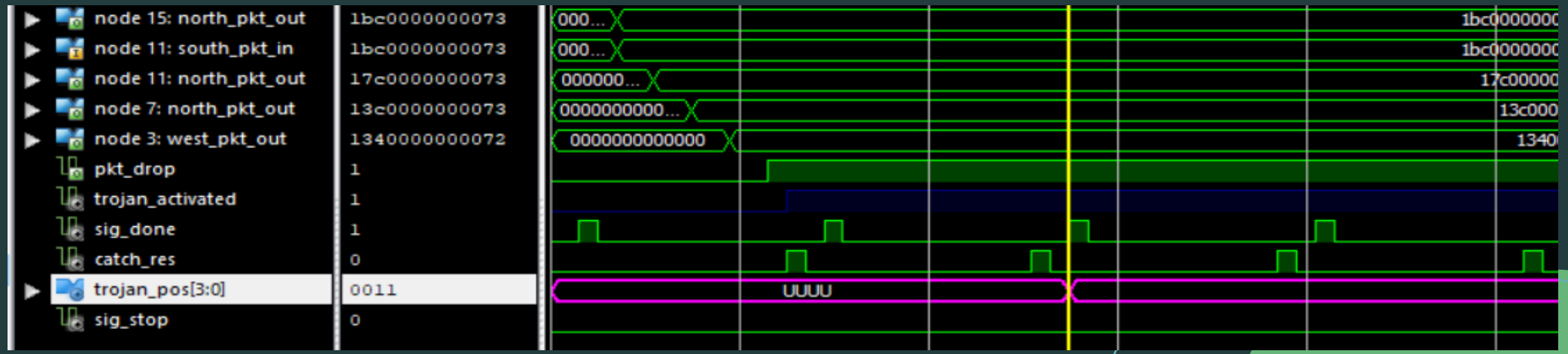
- Path node & Other node

```
Trojan in node 0011(3)
Dark Bule: Packet from 1111(15) to 0000(0), packet dropped in node 2
Sky Bule: Packet from 1101(13) to 0110(6)
```



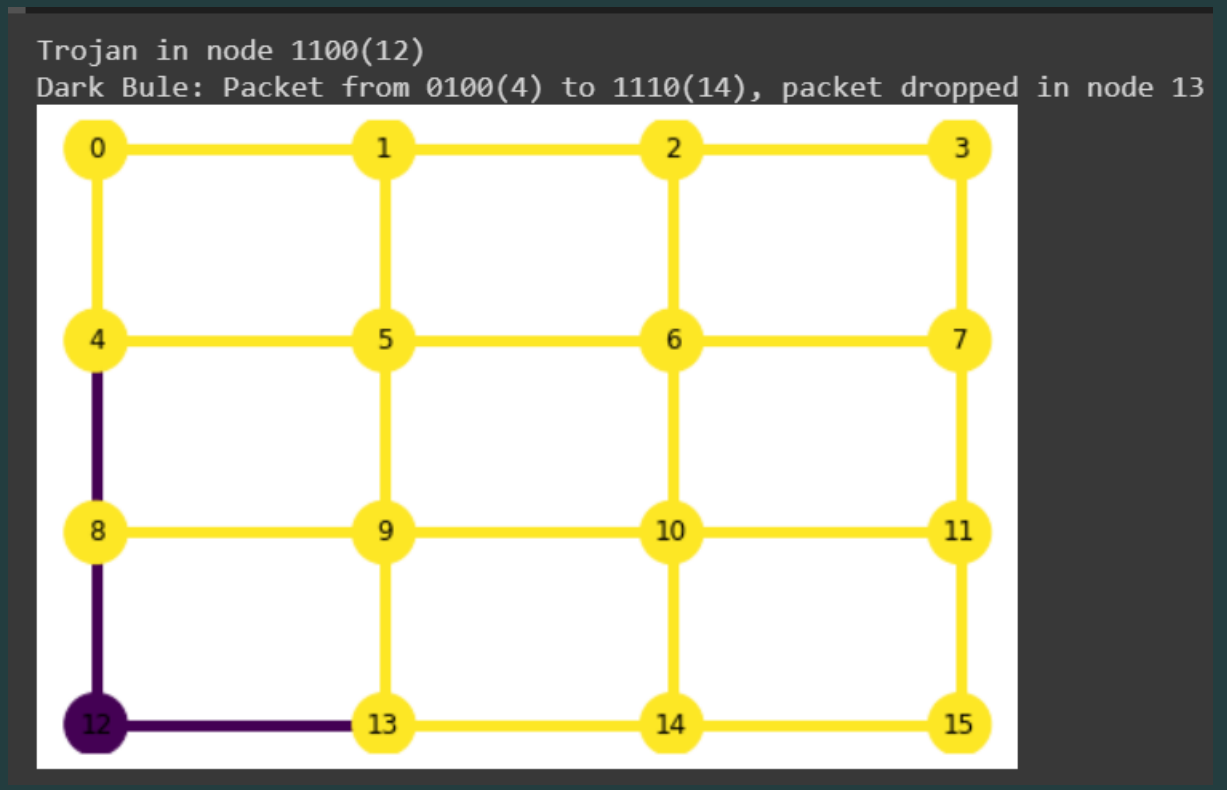
- Sky bule path: packet from node 13 to node 6

- Dark bule path: packet from node 15 to node 0
  - Packet dropped in node 2

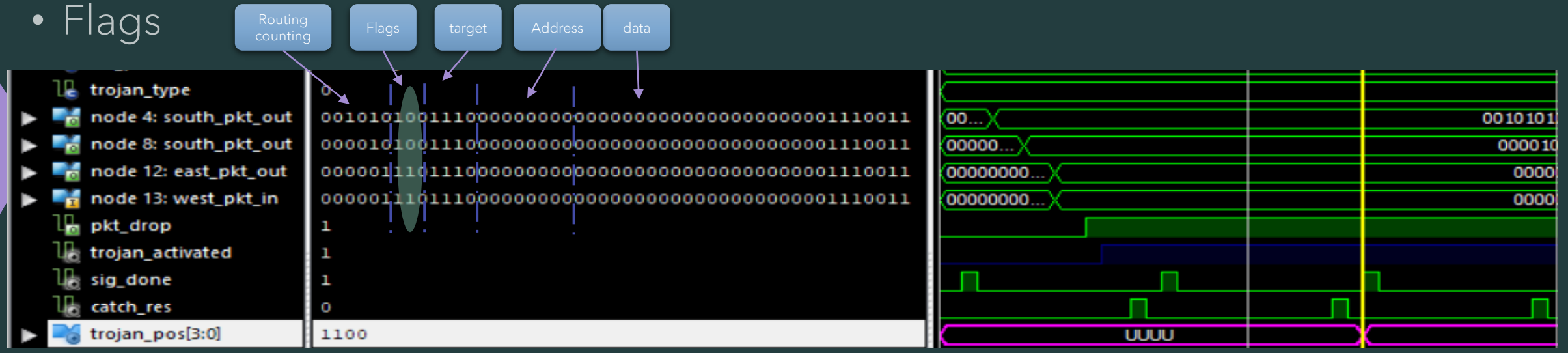


Testing

- Testing 3
- Different type of Trojan
  - Trojan type
    - 0: flags
    - 1: data
    - 2: target id



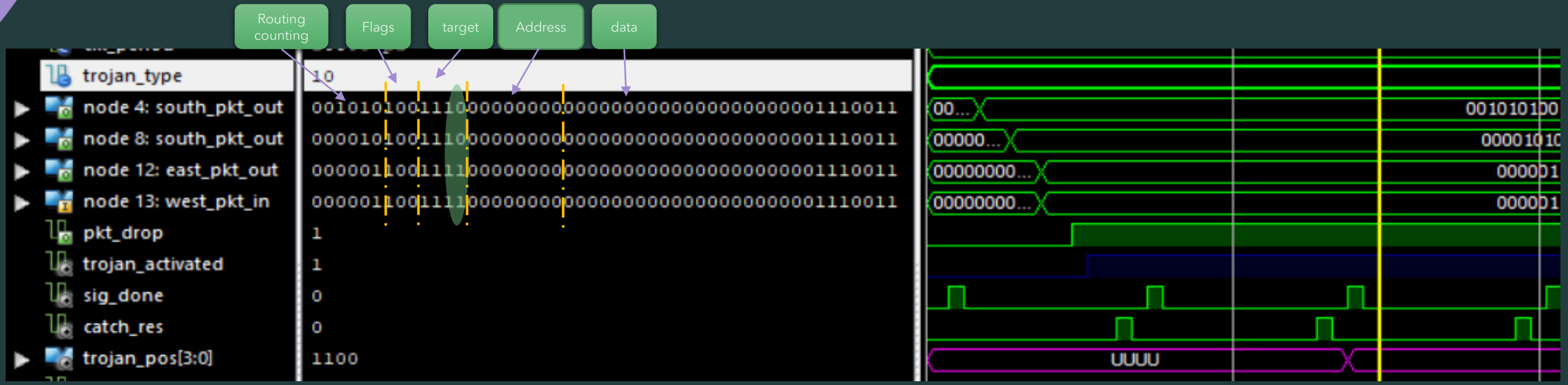
- Flags


Routing counting
Flags
target
Address
data
trojan_type
node 4: south_pkt_out
node 8: south_pkt_out
node 12: east_pkt_out
node 13: west_pkt_in
pkt_drop
trojan_activated
sig_done
catch_res
trojan_pos[3:0]
1100
UUUU

- Data

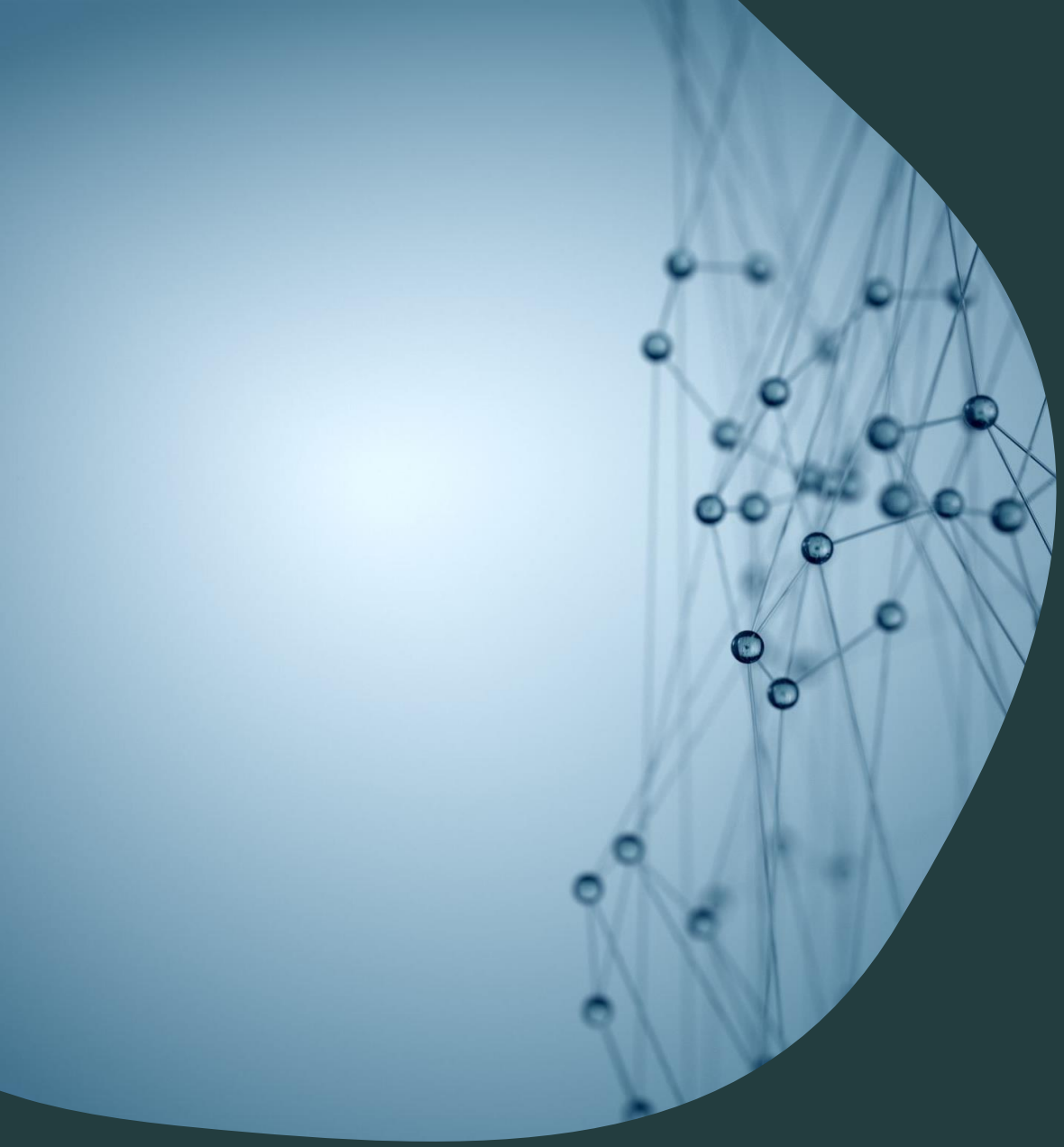

Routing counting
Flags
target
Address
data
trojan_type
node 4: south_pkt_out
node 8: south_pkt_out
node 12: east_pkt_out
node 13: west_pkt_in
pkt_drop
trojan_activated
sig_done
catch_res
trojan_pos[3:0]
1100
UUUU

- Target ID


Routing counting
Flags
target
Address
data
trojan_type
10
node 4: south_pkt_out
node 8: south_pkt_out
node 12: east_pkt_out
node 13: west_pkt_in
pkt_drop
1
trojan_activated
1
sig_done
0
catch_res
0
trojan_pos[3:0]
1100
UUUU

Evaluation

- Compare to the authentication attack detection
  - High accuracy
    - Narrows the attack path
- Compare to the EETD
  - Low overhead
    - Tag segment is not in the packet
    - The tag is different from node to node



Timestamp

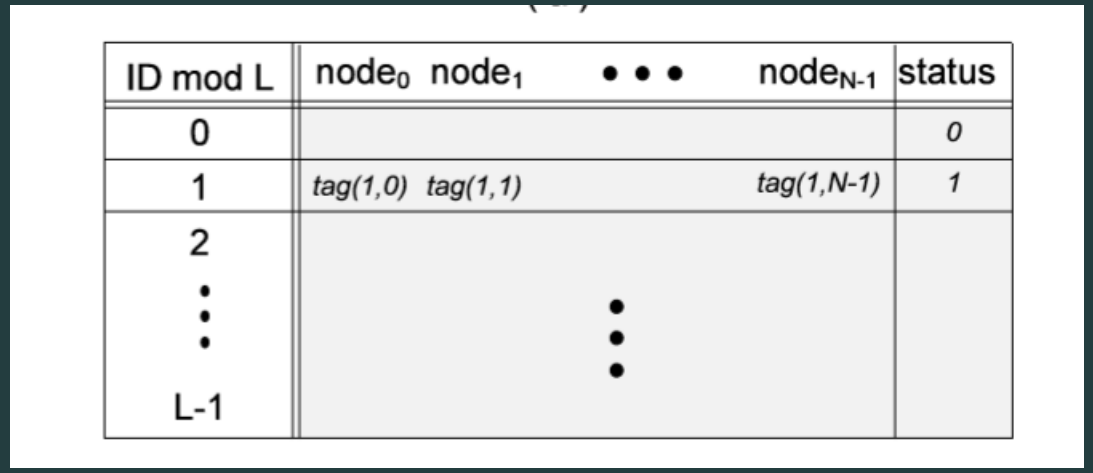
Plan from Thesis B

Task	Plan Duration	Done?
Produce an Improved Design	2 weeks	Yes
The basic Model	2-3 weeks	Yes
Insert Trojan and Check	0.5-1 week	Yes
Progressive Authentication	2-3 weeks	Yes
Make the data closer to the report	1 week	Yes
A simple Improved Design Demo	2-3 weeks	Yes
Enhancement	1-2 weeks	Yes

Tasks	Expected	Actual
Basic Framework	2 weeks	2 weeks
Detection design implementation	2 weeks	2 weeks
Optimization	1 week	1.5 week
Testing	1 week	1 week
Improvement	2 week	currently

Improvement

- Tag table
  - L: maximum L alive packets in NoC
    - Currently: L = 1
  - Status bit: control the use of the tag entries in table
- High level buffers in router – buffer queue
  - In case of the channel is busy
  - A node receives two more packets



ID mod L	$node_0$ $node_1$ • • • $node_{N-1}$	status
0		*0*
1	*tag(1,0) tag(1,1) tag(1,N-1)*	*1*
2 ⋮ L-1	⋮	


Q&A