Discrete-Event Modeling Using DEVS

OpenFlow Protocol (OFP) Simulator

Assignment1

SYSC 5104 - Methodologies for Discrete Event Modelling and Simulations

Fall 2020

Mohammed Al-Saeedi
101109046
Carleton University

TABLE OF CONTENTS

Part I

Overview:

OpenFlow (OF) [1] is a flow control protocol for Software-Defined Networks (SDN) which acts as the southbound interface between the network control plane (Controller) and data plane (data forwarding devices). It allows the controller to have direct control and access to the forwarding devices (e.g., Switch, vSwitche, WSN node). An OpenFlow-based forwarding device consists of flow tables that stores flow entries and a openflow channel to communicate with the controller, as shown in Figure 1. Every flow table has a set of entries that consists of matching fields, statistical counters, and a set of flow instructions (actions). The matching fields are used to forward the packet as per the corresponding entry's action.

The OpenFlow-based forwarding process starts when a new packet arrives at the forwarding device and matches against the stored flow entries. If a matched flow entry is found, the packet will be forwarded to the next routing hop. Otherwise, the packet will be forwarded to the controller as a *packet_in* message. The controller will then calculate the forwarding rules of the new unknown packet header and send a *packet_out* message to install the routing rules as flow entries to all the flow tables of the corresponding forwarding devices along the routing path [2].

The purpose of this project is to simulate the SDN-OpenFlow protocol using Discrete-Event Systems Specification (DEVS) and assess its performance in terms of packet latency and controller overhead (number of packets forwarded to the controller). The SDN-OpenFlow DEVS Simulator consists of four main components: Sender (atomic), OFRouter (coupled), Controller (atomic), and Receiver (atomic) as shown in Figure 2. A controller can be a cluster of controllers working together to perform as one logically centralized control. For simplicity, we assume that there is only one controller. A sender represents an end-user or server that sends a packet to a receiver. The OFRouter is the OpenFlow-based forwarding element (e.g., Switch, vSwitch, WSN node) which can be modeled as a DEVS coupled component consisting of the input buffer, processor, and matching flow table. Due to resource constraints in the forwarding elements, a flow table has a limited set of flow entries and each flow entry consists of matching fields, instructions, and an idle_timeout. The flow entry's idle_timeout is extended if it matches any arrived packet before its expiration. Otherwise, the flow entry will be considered as a passive entry and be removed from the flow table to provide space for other active flow entries. The input and output to/from the OFRouter will be in a tabular structure with different variables.

The routing process starts when a flow packet arrives at the ingress OFRouter. The packet will be queued in an input buffer to be then processed by matching its header against the flow table entries. If a matched flow entry is found, the associated instructions will be executed to forward the node to the next forwarding hop. Otherwise, the packet will be sent to the controller in a packet_in request message to get the appropriate routing information. For simplicity, the controller can be modeled as one entity responsible for calculating and installing the routing rules to the corresponding OFRouters within a non-deterministic delay expressed by a normal distribution with a mean and deviation.

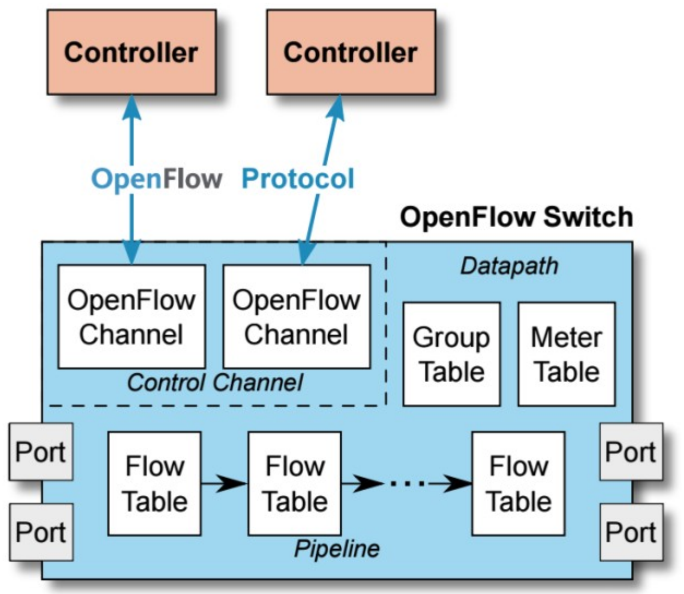


Figure 1: OpenFlow version 1.5 switch components [2]

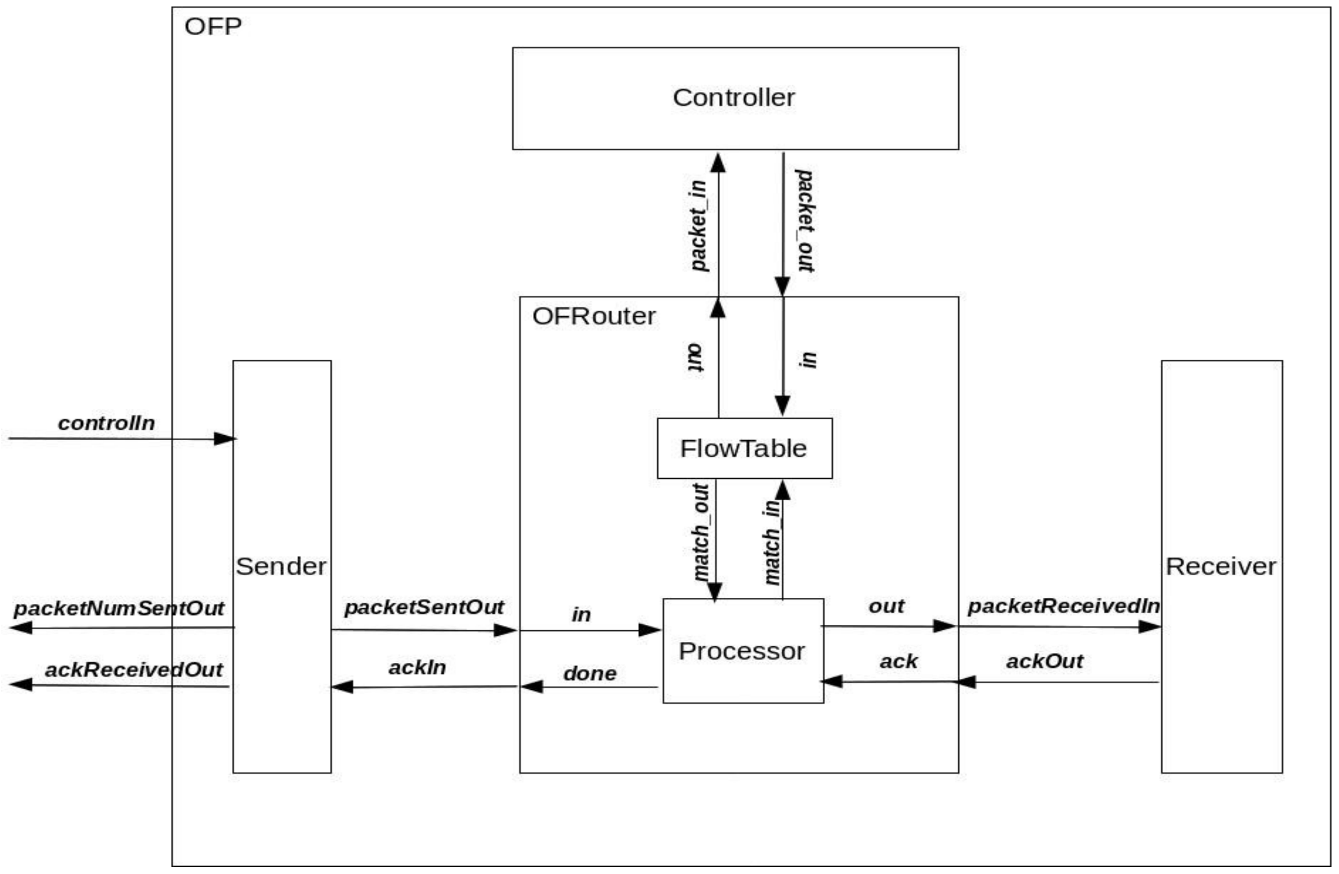


Figure 2: Structure of OFP Simulator

Part II

As shown in Figure 1, the OpenFlow Protocol (OFP) simulator has the controlIn input that initiates the number of packets that should be sent. The OFP Simulator consists of 3 components: Sender, OFRouter, and Receiver. The sender transmits packets to the OpenFlow-supported router (OFRouter) which then forwards the packet to the next OFRouter using the routing instructions in the flow tables. For simplicity, we assume that there is only one router between the sender and the receiver. The sender receives an acknowledgment from the OFRouter. The OFRouter further decomposed into two components: Processor and FlowTable. The pipeline forwarding process is tackled by these two components. However, if no matched routing rule is found in the flowtable, the packet is forwarded to the centralized controller to get its routing path. The matched packet will then be forwarded to its routing destination by the processor.

Formal Specifications:

Packet Structure:

The simulated packet structure consists of three parts: the packetNum, the maching header and a one bit that indicates whether the packet has a matched flow entry.

PacketNum	match	matched

Atomic Models:

The formal specificaitons <S, X, Y, δint, δext, λ, ta> for the atomic models are defined as following:

Sender:

State Variables:

Packet packet;	// the packet generated by the sender.
int totalPacketNum;	//total number of packets to be transmitted.
bool ack;	//true if expected sending-acknowledgement is received.
Int ackNum;	//shows the acknowledged packet
bool sending;	//true for sending the packet.
bool model_active;	//true: active, false: passive
TIME next_internal;	//identifies the next interval

Formal Specification:

X = {controlIn, ackIn}

Y = {packetNumSentOut, packetSentOut, ackReceivedOut}

S = {model_active, next_internal, packet.matched, sending, ack}

```
δint (model_active, packet, sending, ack, e, x){
            case ack
                        true:
            if (packet.packetNum < totalPacketNum){ //send the next packet
                packet.packetNum ++;
                ack = false;
                packet.match = (uint32_t)rand(); //generate random matching header
                sending = true;
                model_active = true;
                next_internal = preparationTime;
                } else { //all packets have been sent out successfully
                    model_active = false;
                    next_internal = INFINITY; //passivate
                }
            } else{
                if (sending){ //timeout before sending again
                    sending = false;
                    model_active = true;
                    next_internal = timeout;
                } else { //timeout elapsed, then re-send the previous packet
                    sending = true;
                    model_active = true;
                    next_internal = preparationTime;    //sending time
                }
            }
}

δext (model_active, packet, sending, ack, e, x){
            for(get x from port controlIn){
                if(model_active == false){//check if the sender is active
                    totalPacketNum = x; //get the number of packets to be sent
                    if (totalPacketNum > 0){
                        packet.packetNum = 1; //sent a new packet
                        packet.match = (uint32_t)rand(); //generate a random matching header
                        ack = false; //packet still not acknowledged
                        sending = true;
                        packet.matched = false; //packet still not matched to get its route
                        model_active = true; //
                        next_internal = preparationTime; //sending time
                    }else{
                        if(next_internal != INFINITY){
                            next_internal = next_internal - e;
                        }
                    }
                }
            }
            for(get x from port ackIn){
                if(model_active == true) { //check whether the sender is active
                    state.ack = true; //packet acknowledged
                    state.ackNum = x; //set the acknowledged packet number
                    state.sending = false; //
                    state.next_internal = 0; //trigger an internal function immediately
                }
            }
}

λ(sending){
    if (sending){
                send_packet_through_packetSentOut_bag;
                send_packetNum_through_packetNumSentOut_bags;
            }else{
                if (state.ack){ //if the packet has been acknowledged
                    send_ackNum_through_ackReceivedOut_bags;
                }
            }
}
```

Processor:

State Variables:

bool transmitting;	//true: for sending the packet.
bool ack;	//true if expected sending-acknowledgement is received.
Int ackNum;	//shows the number of the acknowledged packet
Packet packet;	//packet payload and header.

Formal Specification:

```
X = {in, match_in, ack}
Y = {out, match_out, done}
S = {packet.matched, transmitting, ack}

δint (packet, transmitting, ack, e, x){
        transmitting = false;
}

δext (packet, transmitting, ack, e, x){
        for(get x from port n){
        packet = x; //set the packet
```

```
          ack = false; //the packet is not acknowledged yet
          transmitting = true; //send it to the flowtable to get routing instructions
      }

      for(get x from port ack){
          ackNum = x; //get the number of acknowledged packet
          ack = true; //set ack to true
          transmitting = true; //send the acknowledged packet number to the sender
      }

      for(get x from prot match_in){
          packet.matched = x.matched; //packet has been matched by the flowtable/controller
          ack = false;
          transmitting = true;   //send the matched packet (as it has already the routing instructions) to the
next level (receiver)
      }
}

λ(transmitting, packet.matched, ack){
    if (transmitting){
          if(ack){
             send_packetNum_through_done_bag;
          }else{
             if(packet.matched){
                send_packet_through_out_bag;
             }else{
                send_packet_through_match_out_bag;
             }
          }
      }
}
```

FlowTable:

<u>State Variables:</u>

```
      bool transmitting;          //true: for sending the packet.
      Packet packet;              //packet payload and header.
      int flowTableMaxSize;   //maximum size of the flowtable.
      vector<pair<uint32_t, TIME>> flowTable;      //flowtable data structure (consists of flow entries).
```

<u>Formal Specification:</u>

X = {in, match_in,}

Y = {out, match_out}

S = {packet.matched, transmitting}

```
δint (packet, transmitting, ack, e, x){
        transmitting = false;
}

δext(packet, transmitting, ack, e, x){
        //for loop to remove any timeout flow entry elapsed time - flow entry time <> 20 second
  for(get x from port in){
        //add flow entry to the flow table if controler response with the appropriate flow entry and the
flow table is not full
        if (flowTable.size() < flowTableMaxSize){ //check if the flowtable is full
            packet = x; //set the received packet from the controller
            packet.matched = true; //packet is matched (routing instruction provided by the controller)
            transmitting = true;
            pair<uint32_t, TIME> flowEntry = make_pair(packet.match, TIME()); //create new flow entry
            flowTable.push_back(flowEntry); //install packet routing rules as flow entry to the flow table
        }else{
            flowTable.erase(state.flowTable.begin()); //remove oldest vector (flow entry)
            packet = x; //set the received packet from the controller
            packet.matched = true; //packet is matched (routing instruction provided by the controller)
            transmitting = true;
            pair<uint32_t, TIME> flowEntry = make_pair(packet.match, TIME());//create new flow entry
            flowTable.push_back(flowEntry);//install packet routing rules as flow entry to the flow table
        }
    }
    for(get x from match_in){
        packet = x; //get the received packet from the processor
        for(const auto &y : flowTable){ //loockup the matching entry
            if(state.packet.match == y.first){
                packet.matched = true; //corresponding matching entry has been successfully found
            }
        }
        state.transmitting = true; //send back the matched packet to the processor
    }
}
λ(transmitting, packet.matched){
  if(transmitting){
        if(packet.matched){
            send_matchedPacket_through_out_bag;
        }else{
            send_unmatchedPacket_through_match_out_bag;
        }
    }
}
```

Controller:

State Variables:

```
bool transmitting;      //true: for sending the packet.
Packet packet;          //packet payload and header.
TIME next_internal;     //
```

Formal Specification:

X = {packet_in}

Y = {packet_out}

S = {packet.matched, transmitting}

```
δint (packet, transmitting, e, x){
    transmitting = false;
}

δint (packet, transmitting, e, x){
    vector<Packet> message_port_in;
    message_port_in = get_messge_from_port_packet_in;  //get the received packet
    packet = message_port_in[0]; //get the matched packet
    packet.matched = true;   //set the matching header to true as the controller has a global view to the network and knows the appropriate routing path
    transmitting = true; //send the packet back to the router's flowtable
}

λ(transmitting){
    if(transmitting){
        send_matchedPacket_through_packet_out_bag; //send the matched packet
    }
}
```

Receiver:

State Variables:

```
int ackNum;       //number of the acknowledged packet
bool sending;     //true: for sending the packet
```

Formal Specification:

X = {packetReceivedIn}

Y = {ackOut}

S = {ackNum, sending}

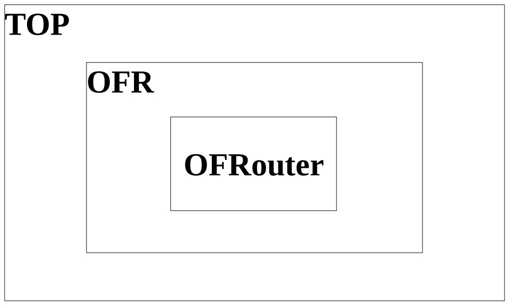
```
δint (packet, sending, e, x){
    sending = false;
}
```

```
δext (packet, sending, e, x){
    vector<Packet> message_port_in;
    message_port_in = get_pakcet_form_port_packetReceivedIn;
    ackNum = message_port_in[0].packetNum; //set the acknowledged packet’s number
    sending = true;   //sending acknowledged packet’s number
}

λ(){
        send_ackNum_through_ackOut_bag; //send the acknowledged packet’s number
}
```

Coupled Models:

The formal specifications <X, Y, D, {M_i}, {I_i}, {Z_{ij}}, SELECT > for the coupled model.



OFRouter:

```
In = {inp_1, inp_2, inp_ack};
Out= {outp_1, outp_2, outp_ack};
SUBMODELS = {processor1, flowtable1};
In_CONNECTIONS = {(inp_1, processor1(in)),
                  (inp_2, flowtable1(in)),
                  (inp_ack, processor1(ack))}
Out_CONNECTIONS = {(processor1(done), outp_ack),
                  (processor1(out), outp_1),
                  (flowtable1(out), outp_2)}
I_CONNECTIONS = {(processor1(match_out), flowtable1(match_in)),
                  (flowtable1(match_out), processor1(match_in))}
```

OFP:

```
In = {inp_control};
Out = {outp, outp_ackNum};
SUBMODELS = {sender1, receiver1, controller1, OFRouter};
In_CONNECTIONS = {(inp_control, Sender(controlIn))}
Out_CONNECTIONS = {(sender1(packetNumSentOut), outp),
                  (sender1(ackReceivedOut), outp_ackNum)}
I_CONNECTIONS = {(sender1(packetSentOut), OFRouter(inp_1)),
                  (receiver1(ackOut), processor1(inp_ack)),
                  (controller1(packet_out), OFRouter(inp_2)),
                  (OFRouter(outp_ack), sender1(ackIn)),
                  (OFRouter(outp_1), receiver1(packetReceivedIn)),
                  (OFRouter(outp_2), controller1(packet_in))}
```

TOP:

```
In = {};
Out= {outp, outp_ackNum};
SUBMODELS = {processor1, flowtable1};
In_CONNECTIONS = {}
Out_CONNECTIONS = {(OFP(out), OFP(out)),
                   (OFP(out_ackNum, out_ackNum)}
I_CONNECTIONS = {(input_reader(out), OFP(inp_control))}
```

Part III

In order to verify the atomic and coupled simulation models, some test scenarios are created to test the OFP simulator.

Test Strategies:

Each atomic or coupled model will be tested as a black box. Any inputs from the adjacent models will be provided in a txt file, so the model will read these inputs and provides its outputs to another text file. Different scenarios will be applied to test the OpenFlow protocol. For simplicity, we use the time unit of seconds which is not reflecting the more-realistic time unit of milliseconds for packet routing.

Test Cases and Execution Analysis

1. Atomic Model Sender:

The sender has two inputs: *controlIn* and *ackIn*. The *controlIn* input is to start up the simulation by providing the number of packets that the sender has to transmit. The number of transmitted packets has to be a positive integer. The sender starts sending packets if and only if it is in a passive mode. When the sender sends a packet it will wait for an acknowledgment message coming from its ackIn port. If the sender receives the acknowledgment of a certain transmitted packet, it will generate a new packet and send it to the router. It will continue doing so until the number of acknowledged packets are equal to the number of packets triggered by the controlIn. If the expected acknowledgment is not received and the timeout expires, the sender will begin to re-send the previous packet. The sending time when the sender begins to send is (00:00:05 time unit) and the timeout for expected acknowledgment is (00:00:15 time unit). Any sent packet will be recorded using the *packetNumSentOut* port to show that the packet has been already sent out.
Any generated packet will be transmitted to the router via *packetSentOut* port. When the sender sent all packets and received their acknowledgments successfully, it returns back to its passive mode.

Shell command: ./SENDER_TEST

Test inputs:
<u>ControlIn Port:</u>
00:00:10 5

<u>ackIn Port:</u>
00:00:17 1
00:00:30 2
00:00:55 3
00:01:05 4
00:02:10 5

Test outputs:

```
00:00:00:000
State for model input_reader_con is next time: 00:00:00:000
State for model input_reader_ack is next time: 00:00:00:000
State for model sender1 is packetNum: 0 & totalPacketNum: 0
00:00:00:000
State for model input_reader_con is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:17:000
State for model sender1 is packetNum: 0 & totalPacketNum: 0
00:00:10:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:00:17:000
State for model sender1 is packetNum: 1 & totalPacketNum: 5
00:00:15:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:00:17:000
State for model sender1 is packetNum: 1 & totalPacketNum: 5
00:00:17:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:00:13:000
State for model sender1 is packetNum: 1 & totalPacketNum: 5
00:00:17:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:00:13:000
State for model sender1 is packetNum: 2 & totalPacketNum: 5
00:00:22:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:00:13:000
State for model sender1 is packetNum: 2 & totalPacketNum: 5
00:00:30:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:00:25:000
State for model sender1 is packetNum: 2 & totalPacketNum: 5
00:00:30:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:00:25:000
State for model sender1 is packetNum: 3 & totalPacketNum: 5
00:00:35:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:00:25:000
State for model sender1 is packetNum: 3 & totalPacketNum: 5
00:00:50:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:00:25:000
State for model sender1 is packetNum: 3 & totalPacketNum: 5
00:00:55:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:00:10:000
State for model sender1 is packetNum: 3 & totalPacketNum: 5
00:00:55:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:00:10:000
State for model sender1 is packetNum: 4 & totalPacketNum: 5
00:01:00:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:00:10:000
State for model sender1 is packetNum: 4 & totalPacketNum: 5
00:01:05:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:01:05:000
State for model sender1 is packetNum: 4 & totalPacketNum: 5
00:01:05:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:01:05:000
State for model sender1 is packetNum: 5 & totalPacketNum: 5
00:01:10:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:01:05:000
State for model sender1 is packetNum: 5 & totalPacketNum: 5
00:01:25:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:01:05:000
State for model sender1 is packetNum: 5 & totalPacketNum: 5
00:01:30:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:01:05:000
State for model sender1 is packetNum: 5 & totalPacketNum: 5
00:01:45:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:01:05:000
State for model sender1 is packetNum: 5 & totalPacketNum: 5
00:01:50:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:01:05:000
State for model sender1 is packetNum: 5 & totalPacketNum: 5
00:02:05:000
```

```
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: 00:01:05:000
State for model sender1 is packetNum: 5 & totalPacketNum: 5
```
00:02:10:000
```
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
00:02:10:000
State for model input_reader_con is next time: inf
State for model input_reader_ack is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
```

The time in bold shows the time when the packet acknowledgment is received by the sender and then it starts sending a new packet.

2. Atomic Model Processor:

The processor atomic model is part of the coupled OFRouter model which represents the router supported by OpenFlow protocol. The processor is responsible for forwarding the packet either to the flowtable or to the receiver. Any received packet to the processor via its *in* port is considered as an unknown packet (has no routing rules). The received packet will then be forwarded directly to the flowtable via its *match_in* port to get its forwarding (routing) instructions (rules). However, if the flowtable doesn’t have the corresponding routing instructions it will then forward it to the controller and get the needed routing rules. The packet will be then send back to the processor with its routing rules (matched packet). The processor forwards the matched-packet to the receiver via its *out* port and waits for an acknowledgment from the receiver from its *ack* port. The processor will not processes any other packet unless it received the acknowledgment of the previous packetNum. When the processor received the acknowledgment message, it will send a done message to the sender and proceed in processing the new packet. The processing time is (00:00:02 time unit).

Shell command: `./PROCESSOR_TEST`

Test inputs:

in Port:
```
00:00:10 1 0 1998989383
00:00:20 2 0 1787289383
00:00:30 3 0 1804289383
00:00:40 4 0 1809889383
00:00:50 5 0 1804988383
```

match_in Port:
```
00:00:13 1 1 1998989383
00:00:23 2 1 1787289383
00:00:34 3 1 1804289383
00:00:43 4 1 1809889383
00:00:54 5 1 1804988383
```

ack Port:
```
00:00:16 1
00:00:27 2
00:00:37 3
00:00:46 4
00:00:58 5
```

Test outputs:
```
00:00:00:000
State for model input_reader_packet is next time: 00:00:00:000
State for model input_reader_ack is next time: 00:00:00:000
State for model input_reader_matchin is next time: 00:00:00:000
State for model processor1 is packetNum: 0 & transmitting: 0
00:00:00:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:16:000
State for model input_reader_matchin is next time: 00:00:13:000
State for model processor1 is packetNum: 0 & transmitting: 0
00:00:10:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:16:000
State for model input_reader_matchin is next time: 00:00:13:000
State for model processor1 is packetNum: 1 & transmitting: 1
00:00:12:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:16:000
State for model input_reader_matchin is next time: 00:00:13:000
State for model processor1 is packetNum: 1 & transmitting: 0
00:00:13:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:16:000
State for model input_reader_matchin is next time: 00:00:10:000
State for model processor1 is packetNum: 1 & transmitting: 1
00:00:15:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:16:000
State for model input_reader_matchin is next time: 00:00:10:000
State for model processor1 is packetNum: 1 & transmitting: 0
00:00:16:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:11:000
State for model input_reader_matchin is next time: 00:00:10:000
State for model processor1 is packetNum: 1 & transmitting: 1
00:00:18:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:11:000
State for model input_reader_matchin is next time: 00:00:10:000
State for model processor1 is packetNum: 1 & transmitting: 0
00:00:20:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:11:000
State for model input_reader_matchin is next time: 00:00:10:000
State for model processor1 is packetNum: 2 & transmitting: 1
00:00:22:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:11:000
State for model input_reader_matchin is next time: 00:00:10:000
State for model processor1 is packetNum: 2 & transmitting: 0
00:00:23:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:11:000
State for model input_reader_matchin is next time: 00:00:11:000
State for model processor1 is packetNum: 2 & transmitting: 1
00:00:25:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:11:000
State for model input_reader_matchin is next time: 00:00:11:000
State for model processor1 is packetNum: 2 & transmitting: 0
00:00:27:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:10:000
State for model input_reader_matchin is next time: 00:00:11:000
State for model processor1 is packetNum: 2 & transmitting: 1
00:00:29:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:10:000
State for model input_reader_matchin is next time: 00:00:11:000
State for model processor1 is packetNum: 2 & transmitting: 0
00:00:30:000
```

```
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:10:000
State for model input_reader_matchin is next time: 00:00:11:000
State for model processor1 is packetNum: 3 & transmitting: 1
00:00:32:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:10:000
State for model input_reader_matchin is next time: 00:00:11:000
State for model processor1 is packetNum: 3 & transmitting: 0
00:00:34:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:10:000
State for model input_reader_matchin is next time: 00:00:09:000
State for model processor1 is packetNum: 3 & transmitting: 1
00:00:36:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:10:000
State for model input_reader_matchin is next time: 00:00:09:000
State for model processor1 is packetNum: 3 & transmitting: 0
00:00:37:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:09:000
State for model input_reader_matchin is next time: 00:00:09:000
State for model processor1 is packetNum: 3 & transmitting: 1
00:00:39:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:09:000
State for model input_reader_matchin is next time: 00:00:09:000
State for model processor1 is packetNum: 3 & transmitting: 0
```
00:00:40:000
```
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:09:000
State for model input_reader_matchin is next time: 00:00:09:000
State for model processor1 is packetNum: 4 & transmitting: 1
00:00:42:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:09:000
State for model input_reader_matchin is next time: 00:00:09:000
State for model processor1 is packetNum: 4 & transmitting: 0
00:00:43:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:09:000
State for model input_reader_matchin is next time: 00:00:11:000
State for model processor1 is packetNum: 4 & transmitting: 1
00:00:45:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:09:000
State for model input_reader_matchin is next time: 00:00:11:000
State for model processor1 is packetNum: 4 & transmitting: 0
00:00:46:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:12:000
State for model input_reader_matchin is next time: 00:00:11:000
State for model processor1 is packetNum: 4 & transmitting: 1
00:00:48:000
State for model input_reader_packet is next time: 00:00:10:000
State for model input_reader_ack is next time: 00:00:12:000
State for model input_reader_matchin is next time: 00:00:11:000
State for model processor1 is packetNum: 4 & transmitting: 0
00:00:50:000
State for model input_reader_packet is next time: inf
State for model input_reader_ack is next time: 00:00:12:000
State for model input_reader_matchin is next time: 00:00:11:000
State for model processor1 is packetNum: 5 & transmitting: 1
00:00:52:000
State for model input_reader_packet is next time: inf
State for model input_reader_ack is next time: 00:00:12:000
State for model input_reader_matchin is next time: 00:00:11:000
State for model processor1 is packetNum: 5 & transmitting: 0
00:00:54:000
State for model input_reader_packet is next time: inf
State for model input_reader_ack is next time: 00:00:12:000
State for model input_reader_matchin is next time: inf
State for model processor1 is packetNum: 5 & transmitting: 1
00:00:56:000
State for model input_reader_packet is next time: inf
State for model input_reader_ack is next time: 00:00:12:000
State for model input_reader_matchin is next time: inf
State for model processor1 is packetNum: 5 & transmitting: 0
00:00:58:000
State for model input_reader_packet is next time: inf
State for model input_reader_ack is next time: inf
State for model input_reader_matchin is next time: inf
State for model processor1 is packetNum: 5 & transmitting: 1
00:01:00:000
State for model input_reader_packet is next time: inf
State for model input_reader_ack is next time: inf
State for model input_reader_matchin is next time: inf
State for model processor1 is packetNum: 5 & transmitting: 0
```

The time unit in bold shows the time when the processor starts processing the newly received packet.

3. Atomic Model FlowTable:

The flowtable is part of the OFRouter coupled model. It is responsible for finding the appropriate routing rules for the arriving flow's packet. The flowtable receives packets from the processor via the match_in port to find their routing rules (instructions). The flow table search for the appropriate routing rules in its stored flow entries. If no matched entry was found, the flowtable forwards the packet via its *out* port to the controller to get its route. However, if a matched flow entry is found, the flowtable will return back the packet to the processor with its routing instructions via *match_out* port. The lookup (matching) process is (00:00:03 time unit). Any time the flowtable received a packet from the controller via its in port, it will store its routing rules as a flow entry with an idle_timeout for future use.

Shell command: ./FLOWTABLE_TEST

Test inputs:

match_in Port: //port that receives packets from the processor to get routing rules from the flowtable
```
00:00:10 1 0 1998989383
00:00:20 2 1 1787289383
00:00:30 3 0 1804289383
00:00:40 4 0 1809889383
00:00:50 5 1 1804988383
```

in Port: //port that receives packets from the controller
```
00:00:15 1 1 1998989383
00:00:35 3 1 1804289383
00:00:45 4 1 1809889383
```

we assume here that the flowtable has the forwarding (routing) rules for packet 2 and 5

Test outputs:
```
00:00:00:000
State for model input_reader_matchQ is next time: 00:00:00:000
State for model input_reader_matchR is next time: 00:00:00:000
State for model flowtable1 is packetNum: 0 & matched: 0
00:00:00:000
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: 00:00:15:000
State for model flowtable1 is packetNum: 0 & matched: 0
00:00:10:000
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: 00:00:15:000
State for model flowtable1 is packetNum: 1 & matched: 0
00:00:13:000
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: 00:00:15:000
State for model flowtable1 is packetNum: 1 & matched: 0
00:00:15:000
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: 00:00:20:000
State for model flowtable1 is packetNum: 1 & matched: 1
00:00:18:000
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: 00:00:20:000
State for model flowtable1 is packetNum: 1 & matched: 1
00:00:20:000
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: 00:00:20:000
State for model flowtable1 is packetNum: 2 & matched: 1
00:00:23:000
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: 00:00:20:000
State for model flowtable1 is packetNum: 2 & matched: 1
00:00:30:000
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: 00:00:20:000
State for model flowtable1 is packetNum: 3 & matched: 0
00:00:33:000
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: 00:00:20:000
State for model flowtable1 is packetNum: 3 & matched: 0
00:00:35:000
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: 00:00:10:000
State for model flowtable1 is packetNum: 3 & matched: 1
00:00:38:000
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: 00:00:10:000
State for model flowtable1 is packetNum: 3 & matched: 1
00:00:40:000
```

```
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: 00:00:10:000
State for model flowtable1 is packetNum: 4 & matched: 0
00:00:43:000
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: 00:00:10:000
State for model flowtable1 is packetNum: 4 & matched: 0
00:00:45:000
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: inf
State for model flowtable1 is packetNum: 4 & matched: 1
00:00:48:000
State for model input_reader_matchQ is next time: 00:00:10:000
State for model input_reader_matchR is next time: inf
State for model flowtable1 is packetNum: 4 & matched: 1
00:00:50:000
State for model input_reader_matchQ is next time: inf
State for model input_reader_matchR is next time: inf
State for model flowtable1 is packetNum: 5 & matched: 1
00:00:53:000
State for model input_reader_matchQ is next time: inf
State for model input_reader_matchR is next time: inf
State for model flowtable1 is packetNum: 5 & matched: 1
```

The results show that neither packet 2 nor packet 5 need to be forwarded to the controller to get the routing rules because the flowtable already has the matched flow entries.

4. Atomic Model Controller:

The controller is an atomic model responsible for calculating the routing rules for any unknown packet forwarded by the OpenFlow-supported router (in the data plane). The controller receives packets from the flowtable via its *packet_in* port and transmits the result via its *packet_out* port. Normally, the control plane can be a cluster of controllers that are logically centralized to act as a single-centralized controller. The response time that a controller needs to calculate the routing rule for an unknown packet can negatively affect the performance of the network. The response time is nondeterministic and better to be simulated as a random variable. However, for simplicity here we will assume that it is deterministic and the controller can calculate the result within (00:00:07 time unit). As future work, it can be simulated as a random variable and test how the controller can affect the performance by calculating the response time, end-to-end delay, and communication overhead due to the number of packets transmitted to the controller. The idle-timeout of the flow entries stored in the flowtable has to be dynamic and adaptively-configured based on the traffic patterns to avoid sending more traffic to the controller and then increase the overall delay. In this testing scenario, we will just test how the controller interacts with the data plane and leave the performance issue for future work.

Shell command: ./CONTROLLER_TEST

Test inputs:

packet_in Port: //port that receives packets from the flowtable to get routing rules from the controller

```
00:00:10 1 0 1998989383
00:00:20 2 0 1787289383
00:00:30 3 0 1804289383
00:00:40 4 0 1809889383
00:00:50 5 0 1804988383
```

Test outputs:

```
00:00:00:000
State for model input_reader_packetIn is next time: 00:00:00:000
State for model controller1 is packetNum: 0 & transmitting: 0
00:00:00:000
State for model input_reader_packetIn is next time: 00:00:10:000
State for model controller1 is packetNum: 0 & transmitting: 0
00:00:10:000
State for model input_reader_packetIn is next time: 00:00:10:000
State for model controller1 is packetNum: 1 & transmitting: 1
00:00:17:000
State for model input_reader_packetIn is next time: 00:00:10:000
State for model controller1 is packetNum: 1 & transmitting: 0
00:00:20:000
State for model input_reader_packetIn is next time: 00:00:10:000
State for model controller1 is packetNum: 2 & transmitting: 1
00:00:27:000
State for model input_reader_packetIn is next time: 00:00:10:000
State for model controller1 is packetNum: 2 & transmitting: 0
00:00:30:000
State for model input_reader_packetIn is next time: 00:00:10:000
State for model controller1 is packetNum: 3 & transmitting: 1
00:00:37:000
State for model input_reader_packetIn is next time: 00:00:10:000
State for model controller1 is packetNum: 3 & transmitting: 0
00:00:40:000
State for model input_reader_packetIn is next time: 00:00:10:000
State for model controller1 is packetNum: 4 & transmitting: 1
00:00:47:000
```

```
State for model input_reader_packetIn is next time: 00:00:10:000
State for model controller1 is packetNum: 4 & transmitting: 0
00:00:50:000
State for model input_reader_packetIn is next time: inf
State for model controller1 is packetNum: 5 & transmitting: 1
00:00:57:000
State for model input_reader_packetIn is next time: inf
State for model controller1 is packetNum: 5 & transmitting: 0
```

5. Atomic Model Receiver:

The flowtable is an atomic model that receives packets from the OFRouter coupling model. It receives packets via its *packetReceivedIn* port and sends acknowledgments via its *ackOut* port. We assume that the receiver sends the acknowledgment packet in a (00:00:02 time unit). If acknowledgment packet will be transmitted until a new packet has been received because this means that the sender has already received the acknowledgment packet for the previous packet.

Shell command: ./RECEIVER_TEST

Test inputs:

packetReceivedIn_Port: //port that receives packets from the processor

```
00:00:10 1 1 1998989383
00:00:20 2 1 1787289383
00:00:30 3 1 1804289383
00:00:40 4 1 1809889383
00:00:50 5 1 1804988383
```

Test outputs:

```
00:00:00:000
State for model input_reader_packet is next time: 00:00:00:000
State for model receiver1 is packetNum: 0
00:00:00:000
State for model input_reader_packet is next time: 00:00:10:000
State for model receiver1 is packetNum: 0
00:00:10:000
State for model input_reader_packet is next time: 00:00:10:000
State for model receiver1 is packetNum: 1
00:00:12:000
State for model input_reader_packet is next time: 00:00:10:000
State for model receiver1 is packetNum: 1
00:00:20:000
State for model input_reader_packet is next time: 00:00:10:000
State for model receiver1 is packetNum: 2
00:00:22:000
State for model input_reader_packet is next time: 00:00:10:000
State for model receiver1 is packetNum: 2
00:00:30:000
State for model input_reader_packet is next time: 00:00:10:000
State for model receiver1 is packetNum: 3
00:00:32:000
State for model input_reader_packet is next time: 00:00:10:000
State for model receiver1 is packetNum: 3
00:00:40:000
State for model input_reader_packet is next time: 00:00:10:000
State for model receiver1 is packetNum: 4
00:00:42:000
State for model input_reader_packet is next time: 00:00:10:000
State for model receiver1 is packetNum: 4
00:00:50:000
State for model input_reader_packet is next time: inf
State for model receiver1 is packetNum: 5
00:00:52:000
State for model input_reader_packet is next time: inf
State for model receiver1 is packetNum: 5
```

5. Top Coupled Model Testing:

The Top coupled model is basically the whole project that contains all the atomic and coupled models. In this testing scenario, we are testing the whole simulator as a black box using an input value of 5 packets via its main controlIn port.

Shell command: `./OFP ../input_data/input_ofp.txt`

Test inputs:

controlIn_Port: //port that triggers the number of packets that should be transmitted by the sender

```
00:00:10 5
```

Test outputs:

```
00:00:00:000
State for model input_reader is next time: 00:00:00:000
State for model sender1 is packetNum: 0 & totalPacketNum: 0
State for model receiver1 is packetNum: 0
State for model controller1 is packetNum: 0 & transmitting: 0
State for model processor1 is packetNum: 261797504 & transmitting: 0
State for model flowtable1 is packetNum: 0 & matched: 0
00:00:00:000
State for model input_reader is next time: 00:00:10:000
State for model sender1 is packetNum: 0 & totalPacketNum: 0
State for model receiver1 is packetNum: 0
State for model controller1 is packetNum: 0 & transmitting: 0
State for model processor1 is packetNum: 261797504 & transmitting: 0
State for model flowtable1 is packetNum: 0 & matched: 0
00:00:10:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 1 & totalPacketNum: 5
State for model receiver1 is packetNum: 0
State for model controller1 is packetNum: 0 & transmitting: 0
State for model processor1 is packetNum: 261797504 & transmitting: 0
State for model flowtable1 is packetNum: 0 & matched: 0
00:00:15:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 1 & totalPacketNum: 5
State for model receiver1 is packetNum: 0
State for model controller1 is packetNum: 0 & transmitting: 0
State for model processor1 is packetNum: 1 & transmitting: 1
State for model flowtable1 is packetNum: 0 & matched: 0
00:00:17:000
```

```
State for model input_reader is next time: inf
State for model sender1 is packetNum: 1 & totalPacketNum: 5
State for model receiver1 is packetNum: 0
State for model controller1 is packetNum: 0 & transmitting: 0
State for model processor1 is packetNum: 1 & transmitting: 0
State for model flowtable1 is packetNum: 1 & matched: 0
00:00:20:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 1 & totalPacketNum: 5
State for model receiver1 is packetNum: 0
State for model controller1 is packetNum: 1 & transmitting: 1
State for model processor1 is packetNum: 1 & transmitting: 0
State for model flowtable1 is packetNum: 1 & matched: 0
00:00:27:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 1 & totalPacketNum: 5
State for model receiver1 is packetNum: 0
State for model controller1 is packetNum: 1 & transmitting: 0
State for model processor1 is packetNum: 1 & transmitting: 0
State for model flowtable1 is packetNum: 1 & matched: 1
00:00:30:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 1 & totalPacketNum: 5
State for model receiver1 is packetNum: 0
State for model controller1 is packetNum: 1 & transmitting: 0
State for model processor1 is packetNum: 1 & transmitting: 1
State for model flowtable1 is packetNum: 1 & matched: 1
00:00:32:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 1 & totalPacketNum: 5
State for model receiver1 is packetNum: 1
State for model controller1 is packetNum: 1 & transmitting: 0
State for model processor1 is packetNum: 1 & transmitting: 0
State for model flowtable1 is packetNum: 1 & matched: 1
00:00:34:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 1 & totalPacketNum: 5
State for model receiver1 is packetNum: 1
State for model controller1 is packetNum: 1 & transmitting: 0
State for model processor1 is packetNum: 1 & transmitting: 1
State for model flowtable1 is packetNum: 1 & matched: 1
00:00:35:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 1 & totalPacketNum: 5
State for model receiver1 is packetNum: 1
State for model controller1 is packetNum: 1 & transmitting: 0
State for model processor1 is packetNum: 1 & transmitting: 1
State for model flowtable1 is packetNum: 1 & matched: 1
00:00:37:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 1 & totalPacketNum: 5
State for model receiver1 is packetNum: 1
State for model controller1 is packetNum: 1 & transmitting: 0
State for model processor1 is packetNum: 1 & transmitting: 0
State for model flowtable1 is packetNum: 1 & matched: 1
00:00:40:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 1 & totalPacketNum: 5
State for model receiver1 is packetNum: 1
State for model controller1 is packetNum: 1 & transmitting: 0
State for model processor1 is packetNum: 1 & transmitting: 1
State for model flowtable1 is packetNum: 1 & matched: 1
00:00:42:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 1 & totalPacketNum: 5
State for model receiver1 is packetNum: 1
State for model controller1 is packetNum: 1 & transmitting: 0
State for model processor1 is packetNum: 1 & transmitting: 0
State for model flowtable1 is packetNum: 1 & matched: 1
00:00:44:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 1 & totalPacketNum: 5
State for model receiver1 is packetNum: 1
State for model controller1 is packetNum: 1 & transmitting: 0
State for model processor1 is packetNum: 1 & transmitting: 1
State for model flowtable1 is packetNum: 1 & matched: 1
00:00:46:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 1 & totalPacketNum: 5
State for model receiver1 is packetNum: 1
State for model controller1 is packetNum: 1 & transmitting: 0
State for model processor1 is packetNum: 1 & transmitting: 0
State for model flowtable1 is packetNum: 1 & matched: 1
00:00:46:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 2 & totalPacketNum: 5
State for model receiver1 is packetNum: 1
State for model controller1 is packetNum: 1 & transmitting: 0
State for model processor1 is packetNum: 1 & transmitting: 0
State for model flowtable1 is packetNum: 1 & matched: 1
00:00:51:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 2 & totalPacketNum: 5
State for model receiver1 is packetNum: 1
State for model controller1 is packetNum: 1 & transmitting: 0
State for model processor1 is packetNum: 2 & transmitting: 1
State for model flowtable1 is packetNum: 1 & matched: 1
00:00:53:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 2 & totalPacketNum: 5
State for model receiver1 is packetNum: 1
State for model controller1 is packetNum: 1 & transmitting: 0
State for model processor1 is packetNum: 2 & transmitting: 0
State for model flowtable1 is packetNum: 2 & matched: 0
00:00:56:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 2 & totalPacketNum: 5
State for model receiver1 is packetNum: 1
State for model controller1 is packetNum: 2 & transmitting: 1
State for model processor1 is packetNum: 2 & transmitting: 0
State for model flowtable1 is packetNum: 2 & matched: 0
00:01:03:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 2 & totalPacketNum: 5
State for model receiver1 is packetNum: 1
State for model controller1 is packetNum: 2 & transmitting: 0
State for model processor1 is packetNum: 2 & transmitting: 0
State for model flowtable1 is packetNum: 2 & matched: 1
00:01:06:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 2 & totalPacketNum: 5
State for model receiver1 is packetNum: 1
State for model controller1 is packetNum: 2 & transmitting: 0
State for model processor1 is packetNum: 2 & transmitting: 1
State for model flowtable1 is packetNum: 2 & matched: 1
00:01:08:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 2 & totalPacketNum: 5
State for model receiver1 is packetNum: 2
State for model controller1 is packetNum: 2 & transmitting: 0
State for model processor1 is packetNum: 2 & transmitting: 0
State for model flowtable1 is packetNum: 2 & matched: 1
00:01:10:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 2 & totalPacketNum: 5
State for model receiver1 is packetNum: 2
State for model controller1 is packetNum: 2 & transmitting: 0
State for model processor1 is packetNum: 2 & transmitting: 1
State for model flowtable1 is packetNum: 2 & matched: 1
00:01:11:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 2 & totalPacketNum: 5
State for model receiver1 is packetNum: 2
State for model controller1 is packetNum: 2 & transmitting: 0
State for model processor1 is packetNum: 2 & transmitting: 1
State for model flowtable1 is packetNum: 2 & matched: 1
00:01:13:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 2 & totalPacketNum: 5
State for model receiver1 is packetNum: 2
State for model controller1 is packetNum: 2 & transmitting: 0
State for model processor1 is packetNum: 2 & transmitting: 0
State for model flowtable1 is packetNum: 2 & matched: 1
00:01:16:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 2 & totalPacketNum: 5
State for model receiver1 is packetNum: 2
State for model controller1 is packetNum: 2 & transmitting: 0
State for model processor1 is packetNum: 2 & transmitting: 1
State for model flowtable1 is packetNum: 2 & matched: 1
00:01:18:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 2 & totalPacketNum: 5
State for model receiver1 is packetNum: 2
State for model controller1 is packetNum: 2 & transmitting: 0
State for model processor1 is packetNum: 2 & transmitting: 0
State for model flowtable1 is packetNum: 2 & matched: 1
00:01:20:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 2 & totalPacketNum: 5
State for model receiver1 is packetNum: 2
State for model controller1 is packetNum: 2 & transmitting: 0
State for model processor1 is packetNum: 2 & transmitting: 1
State for model flowtable1 is packetNum: 2 & matched: 1
00:01:22:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 2 & totalPacketNum: 5
State for model receiver1 is packetNum: 2
State for model controller1 is packetNum: 2 & transmitting: 0
State for model processor1 is packetNum: 2 & transmitting: 0
State for model flowtable1 is packetNum: 2 & matched: 1
00:01:22:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 3 & totalPacketNum: 5
State for model receiver1 is packetNum: 2
State for model controller1 is packetNum: 2 & transmitting: 0
State for model processor1 is packetNum: 2 & transmitting: 0
State for model flowtable1 is packetNum: 2 & matched: 1
00:01:27:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 3 & totalPacketNum: 5
State for model receiver1 is packetNum: 2
State for model controller1 is packetNum: 2 & transmitting: 0
State for model processor1 is packetNum: 3 & transmitting: 1
State for model flowtable1 is packetNum: 2 & matched: 1
00:01:29:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 3 & totalPacketNum: 5
State for model receiver1 is packetNum: 2
State for model controller1 is packetNum: 2 & transmitting: 0
State for model processor1 is packetNum: 3 & transmitting: 0
State for model flowtable1 is packetNum: 3 & matched: 0
00:01:32:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 3 & totalPacketNum: 5
State for model receiver1 is packetNum: 2
State for model controller1 is packetNum: 3 & transmitting: 1
State for model processor1 is packetNum: 3 & transmitting: 0
State for model flowtable1 is packetNum: 3 & matched: 0
00:01:39:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 3 & totalPacketNum: 5
State for model receiver1 is packetNum: 2
State for model controller1 is packetNum: 3 & transmitting: 0
State for model processor1 is packetNum: 3 & transmitting: 0
State for model flowtable1 is packetNum: 3 & matched: 1
00:01:42:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 3 & totalPacketNum: 5
State for model receiver1 is packetNum: 2
State for model controller1 is packetNum: 3 & transmitting: 0
State for model processor1 is packetNum: 3 & transmitting: 1
State for model flowtable1 is packetNum: 3 & matched: 1
00:01:44:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 3 & totalPacketNum: 5
State for model receiver1 is packetNum: 3
State for model controller1 is packetNum: 3 & transmitting: 0
State for model processor1 is packetNum: 3 & transmitting: 0
State for model flowtable1 is packetNum: 3 & matched: 1
00:01:46:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 3 & totalPacketNum: 5
State for model receiver1 is packetNum: 3
State for model controller1 is packetNum: 3 & transmitting: 0
State for model processor1 is packetNum: 3 & transmitting: 1
State for model flowtable1 is packetNum: 3 & matched: 1
00:01:47:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 3 & totalPacketNum: 5
State for model receiver1 is packetNum: 3
State for model controller1 is packetNum: 3 & transmitting: 0
State for model processor1 is packetNum: 3 & transmitting: 1
State for model flowtable1 is packetNum: 3 & matched: 1
00:01:49:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 3 & totalPacketNum: 5
State for model receiver1 is packetNum: 3
State for model controller1 is packetNum: 3 & transmitting: 0
State for model processor1 is packetNum: 3 & transmitting: 0
State for model flowtable1 is packetNum: 3 & matched: 1
00:01:52:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 3 & totalPacketNum: 5
State for model receiver1 is packetNum: 3
State for model controller1 is packetNum: 3 & transmitting: 0
State for model processor1 is packetNum: 3 & transmitting: 1
State for model flowtable1 is packetNum: 3 & matched: 1
```

```
00:01:54:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 3 & totalPacketNum: 5
State for model receiver1 is packetNum: 3
State for model controller1 is packetNum: 3 & transmitting: 0
State for model processor1 is packetNum: 3 & transmitting: 0
State for model flowtable1 is packetNum: 3 & matched: 1
00:01:56:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 3 & totalPacketNum: 5
State for model receiver1 is packetNum: 3
State for model controller1 is packetNum: 3 & transmitting: 0
State for model processor1 is packetNum: 3 & transmitting: 1
State for model flowtable1 is packetNum: 3 & matched: 1
00:01:58:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 3 & totalPacketNum: 5
State for model receiver1 is packetNum: 3
State for model controller1 is packetNum: 3 & transmitting: 0
State for model processor1 is packetNum: 3 & transmitting: 0
State for model flowtable1 is packetNum: 3 & matched: 1
00:01:58:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 4 & totalPacketNum: 5
State for model receiver1 is packetNum: 3
State for model controller1 is packetNum: 3 & transmitting: 0
State for model processor1 is packetNum: 3 & transmitting: 0
State for model flowtable1 is packetNum: 3 & matched: 1
00:02:03:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 4 & totalPacketNum: 5
State for model receiver1 is packetNum: 3
State for model controller1 is packetNum: 3 & transmitting: 0
State for model processor1 is packetNum: 4 & transmitting: 1
State for model flowtable1 is packetNum: 3 & matched: 1
00:02:05:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 4 & totalPacketNum: 5
State for model receiver1 is packetNum: 3
State for model controller1 is packetNum: 3 & transmitting: 0
State for model processor1 is packetNum: 4 & transmitting: 0
State for model flowtable1 is packetNum: 4 & matched: 0
00:02:08:000
```

```
State for model input_reader is next time: inf
State for model sender1 is packetNum: 4 & totalPacketNum: 5
State for model receiver1 is packetNum: 3
State for model controller1 is packetNum: 4 & transmitting: 1
State for model processor1 is packetNum: 4 & transmitting: 0
State for model flowtable1 is packetNum: 4 & matched: 0
00:02:15:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 4 & totalPacketNum: 5
State for model receiver1 is packetNum: 3
State for model controller1 is packetNum: 4 & transmitting: 0
State for model processor1 is packetNum: 4 & transmitting: 0
State for model flowtable1 is packetNum: 4 & matched: 1
00:02:18:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 4 & totalPacketNum: 5
State for model receiver1 is packetNum: 3
State for model controller1 is packetNum: 4 & transmitting: 0
State for model processor1 is packetNum: 4 & transmitting: 1
State for model flowtable1 is packetNum: 4 & matched: 1
00:02:20:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 4 & totalPacketNum: 5
State for model receiver1 is packetNum: 4
State for model controller1 is packetNum: 4 & transmitting: 0
State for model processor1 is packetNum: 4 & transmitting: 0
State for model flowtable1 is packetNum: 4 & matched: 1
00:02:22:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 4 & totalPacketNum: 5
State for model receiver1 is packetNum: 4
State for model controller1 is packetNum: 4 & transmitting: 0
State for model processor1 is packetNum: 4 & transmitting: 1
State for model flowtable1 is packetNum: 4 & matched: 1
00:02:23:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 4 & totalPacketNum: 5
State for model receiver1 is packetNum: 4
State for model controller1 is packetNum: 4 & transmitting: 0
State for model processor1 is packetNum: 4 & transmitting: 1
State for model flowtable1 is packetNum: 4 & matched: 1
00:02:25:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 4 & totalPacketNum: 5
State for model receiver1 is packetNum: 4
State for model controller1 is packetNum: 4 & transmitting: 0
State for model processor1 is packetNum: 4 & transmitting: 0
State for model flowtable1 is packetNum: 4 & matched: 1
00:02:28:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 4 & totalPacketNum: 5
State for model receiver1 is packetNum: 4
State for model controller1 is packetNum: 4 & transmitting: 0
State for model processor1 is packetNum: 4 & transmitting: 1
State for model flowtable1 is packetNum: 4 & matched: 1
00:02:30:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 4 & totalPacketNum: 5
State for model receiver1 is packetNum: 4
State for model controller1 is packetNum: 4 & transmitting: 0
State for model processor1 is packetNum: 4 & transmitting: 0
State for model flowtable1 is packetNum: 4 & matched: 1
00:02:32:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 4 & totalPacketNum: 5
State for model receiver1 is packetNum: 4
State for model controller1 is packetNum: 4 & transmitting: 0
State for model processor1 is packetNum: 4 & transmitting: 1
State for model flowtable1 is packetNum: 4 & matched: 1
00:02:34:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 4 & totalPacketNum: 5
State for model receiver1 is packetNum: 4
State for model controller1 is packetNum: 4 & transmitting: 0
State for model processor1 is packetNum: 4 & transmitting: 0
State for model flowtable1 is packetNum: 4 & matched: 1
00:02:34:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 4
State for model controller1 is packetNum: 4 & transmitting: 0
State for model processor1 is packetNum: 4 & transmitting: 0
State for model flowtable1 is packetNum: 4 & matched: 1
00:02:39:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 4
State for model controller1 is packetNum: 4 & transmitting: 0
State for model processor1 is packetNum: 5 & transmitting: 1
State for model flowtable1 is packetNum: 4 & matched: 1
00:02:41:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 4
State for model controller1 is packetNum: 4 & transmitting: 0
State for model processor1 is packetNum: 5 & transmitting: 0
State for model flowtable1 is packetNum: 5 & matched: 0
00:02:44:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 4
State for model controller1 is packetNum: 5 & transmitting: 1
State for model processor1 is packetNum: 5 & transmitting: 0
State for model flowtable1 is packetNum: 5 & matched: 0
00:02:51:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 4
State for model controller1 is packetNum: 5 & transmitting: 0
State for model processor1 is packetNum: 5 & transmitting: 0
State for model flowtable1 is packetNum: 5 & matched: 1
00:02:54:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 4
State for model controller1 is packetNum: 5 & transmitting: 0
State for model processor1 is packetNum: 5 & transmitting: 1
State for model flowtable1 is packetNum: 5 & matched: 1
00:02:56:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 5
State for model controller1 is packetNum: 5 & transmitting: 0
State for model processor1 is packetNum: 5 & transmitting: 0
State for model flowtable1 is packetNum: 5 & matched: 1
00:02:58:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 5
State for model controller1 is packetNum: 5 & transmitting: 0
State for model processor1 is packetNum: 5 & transmitting: 1
State for model flowtable1 is packetNum: 5 & matched: 1
00:02:59:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 5
State for model controller1 is packetNum: 5 & transmitting: 0
State for model processor1 is packetNum: 5 & transmitting: 1
State for model flowtable1 is packetNum: 5 & matched: 1
00:03:01:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 5
State for model controller1 is packetNum: 5 & transmitting: 0
State for model processor1 is packetNum: 5 & transmitting: 0
State for model flowtable1 is packetNum: 5 & matched: 1
00:03:04:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 5
State for model controller1 is packetNum: 5 & transmitting: 0
State for model processor1 is packetNum: 5 & transmitting: 1
State for model flowtable1 is packetNum: 5 & matched: 1
00:03:06:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 5
State for model controller1 is packetNum: 5 & transmitting: 0
State for model processor1 is packetNum: 5 & transmitting: 0
State for model flowtable1 is packetNum: 5 & matched: 1
00:03:08:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 5
State for model controller1 is packetNum: 5 & transmitting: 0
State for model processor1 is packetNum: 5 & transmitting: 1
State for model flowtable1 is packetNum: 5 & matched: 1
00:03:10:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 5
State for model controller1 is packetNum: 5 & transmitting: 0
State for model processor1 is packetNum: 5 & transmitting: 0
State for model flowtable1 is packetNum: 5 & matched: 1
00:03:10:000
State for model input_reader is next time: inf
State for model sender1 is packetNum: 5 & totalPacketNum: 5
State for model receiver1 is packetNum: 5
State for model controller1 is packetNum: 5 & transmitting: 0
State for model processor1 is packetNum: 5 & transmitting: 0
State for model flowtable1 is packetNum: 5 & matched: 1
```

References:

[1] A tool for DEVS Modeling and Simulation User’s Guide, Cristina Ruiz Martin Gabriel A. Wainer, Carleton University.

[2] OpenFlow Switch Specification 1.5.0, Standard ONF TS-020, 2013, pp. 1–205.

[3] M. Alsaeedi, M. M. Mohamad and A. A. Al-Roubaiey, "Toward Adaptive and Scalable OpenFlow-SDN Flow Control: A Survey," in *IEEE Access*, vol. 7, pp. 107346-107379, 2019, doi: 10.1109/ACCESS.2019.2932422.