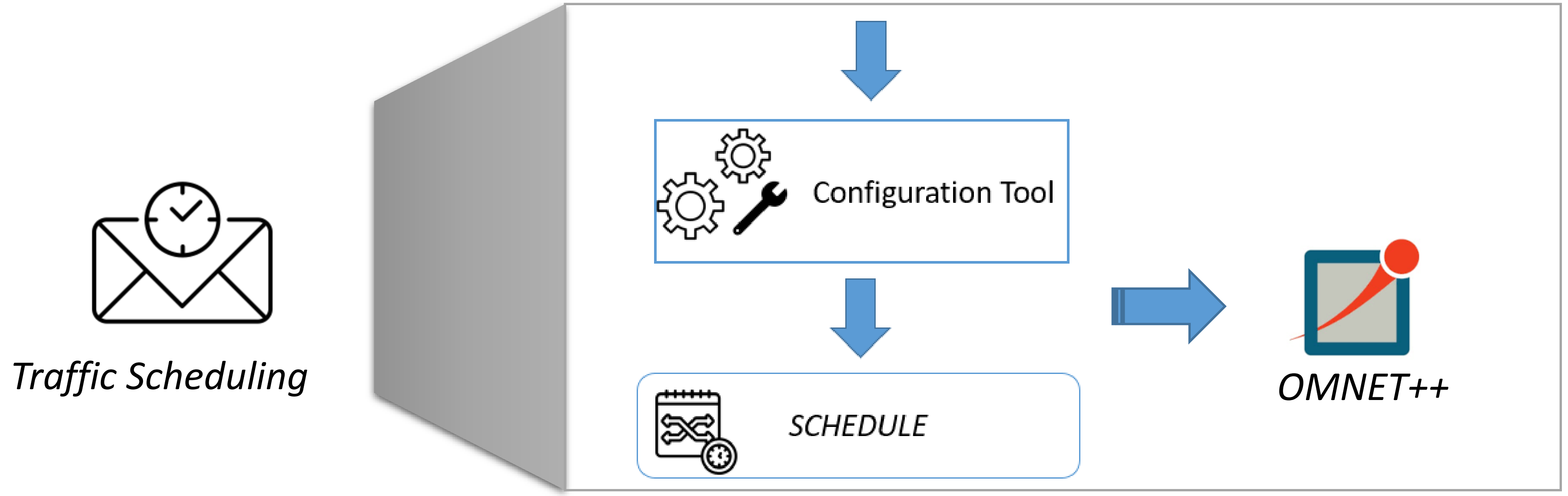
Time Sensitive Network | *Traffic Scheduling Applying Genetic Algorithm*

Title



Abhilash G., Research Scholar
Amrita Vishwa Vidyapeetham, Amritapuri Campus

Agenda

Source: All Icons courtesy: https://thenounproject.com/

Steps| *Overall Approach*

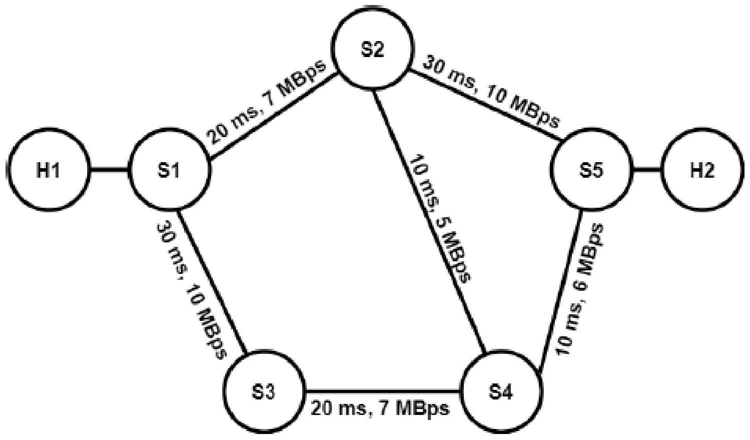
Network Topology Discovery (NED file) 1

Specify the flows and details 2

1. *Flow Name*
2. *Deadline*
3. *Source*
4. *Destination*
5. *Size*

Example:
Video (65KB Frames), "video" , deadline: 200ms

Best Effort (100B)
"best effort", deadline 500ms



Example Topology

2 Configuration Tool

SCHEDULE
Omnet.ini file 3

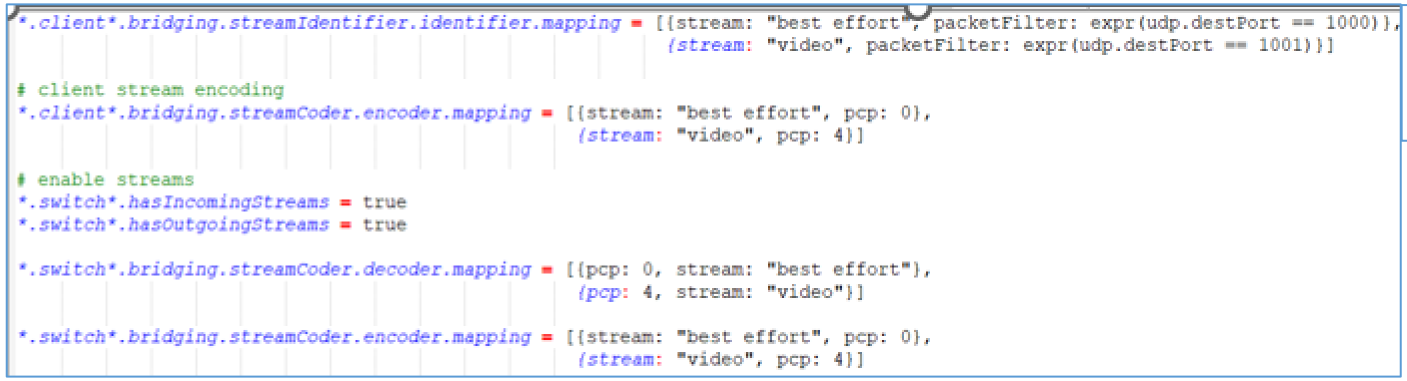
1. *Priority and Flow Identification*
2. *Gate Schedule*
3. *Time period of flows*

Can be used with OMNET++ as part of .ini file

Our approach of Scheduling

TSN| *Omnet.ini file in OMNET++*

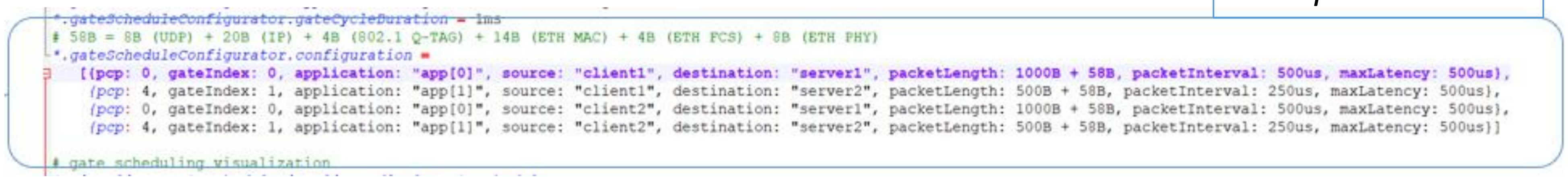
Title



```
*.client*.bridging.streamIdentifier.identifier.mapping = [{stream: "best effort", packetFilter: expr(udp.destPort == 1000)},
                                                         {stream: "video", packetFilter: expr(udp.destPort == 1001)}]

# client stream encoding
*.client*.bridging.streamCoder.encoder.mapping = [{stream: "best effort", pcp: 0},
                                                  {stream: "video", pcp: 4}]

# enable streams
*.switch*.hasIncomingStreams = true
*.switch*.hasOutgoingStreams = true

*.switch*.bridging.streamCoder.decoder.mapping = [{pcp: 0, stream: "best effort"},
                                                  {pcp: 4, stream: "video"}]

*.switch*.bridging.streamCoder.encoder.mapping = [{stream: "best effort", pcp: 0},
                                                  {stream: "video", pcp: 4}]
```

Stream Identification and Mapping (Priorities)

Example Schedule



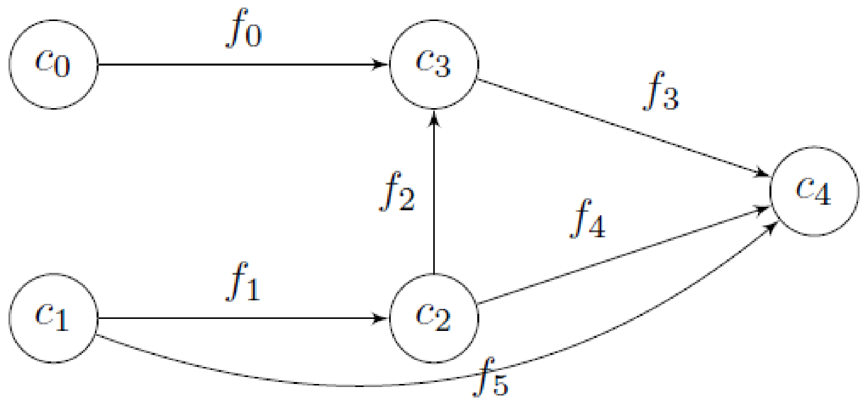
```
*.gateScheduleConfigurator.gateCycleDuration = 1ms
# 58B = 8B (UDP) + 20B (IP) + 4B (802.1 Q-TAG) + 14B (ETH MAC) + 4B (ETH FCS) + 8B (ETH PHY)
*.gateScheduleConfigurator.configuration =
   [{pcp: 0, gateIndex: 0, application: "app[0]", source: "client1", destination: "server1", packetLength: 1000B + 58B, packetInterval: 500us, maxLatency: 500us},
    {pcp: 4, gateIndex: 1, application: "app[1]", source: "client1", destination: "server2", packetLength: 500B + 58B, packetInterval: 250us, maxLatency: 500us},
    {pcp: 0, gateIndex: 0, application: "app[0]", source: "client2", destination: "server1", packetLength: 1000B + 58B, packetInterval: 500us, maxLatency: 500us},
    {pcp: 4, gateIndex: 1, application: "app[1]", source: "client2", destination: "server2", packetLength: 500B + 58B, packetInterval: 250us, maxLatency: 500us}]

# gate scheduling visualization
```

Gate cycle = LCM of all flow periods

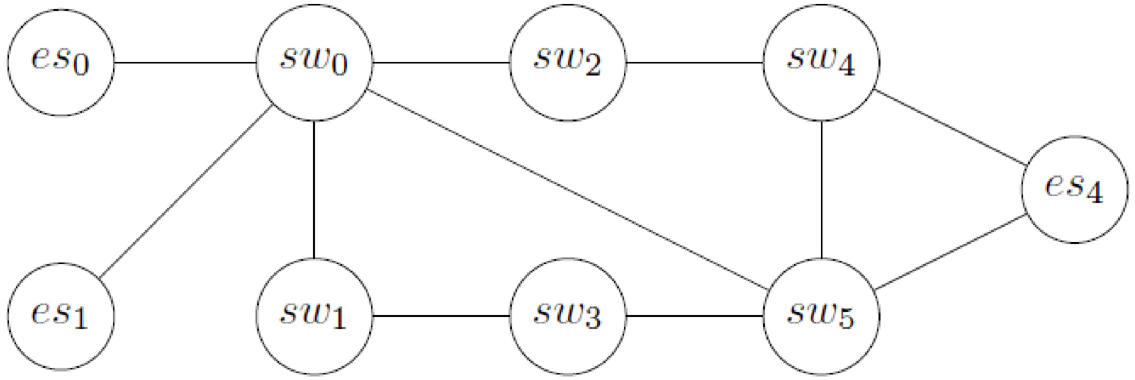
Dictates the flow for each stream

1. Best Effort is PCP:0 and gateIndex - 1 means gate open (0 closed) , source and destination, packetLength, interval and maximumLatency are defined

System Model| *GA Approach*



(a) Application graph



(b) Architecture graph

Fig. 1: An example of system model

GA Aspects| *Calculations and Fitness Function*

$$t_{\text{f.processing}} = ProcessingRate_{\text{device}} \times f.size$$

$$l_i \in E_1 : f_{l_i}.TransDelay = \frac{f.size}{l_i.bandwidth}$$

$$f_{l_{i+1}}.InjecTime = f_{l_i}.InjecTime + t_{\text{f.processing}} +$$

$$f_{l_i}.TransDelay$$

$$\overline{f}.e2eDelay = \sum_{l \in \overline{f}.route} t_{\overline{f}.\text{processing}} + \overline{f}.TransDelay$$

$$\overline{f}.InjecTime + \overline{f}.e2eDelay + c.ExTime$$

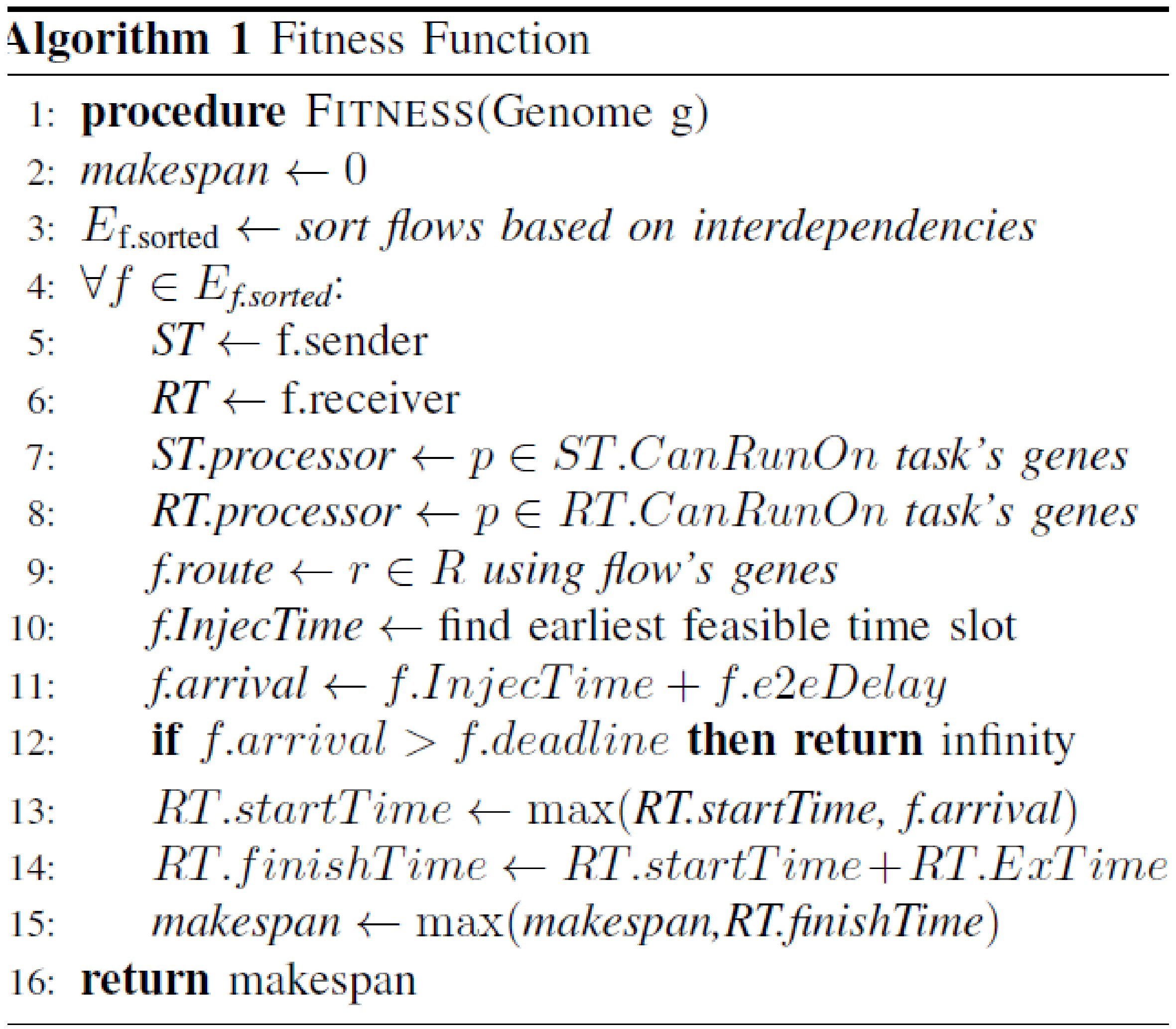
$$\leq f.InjectTime$$

$$f.InjecTime + f.e2eDelay \leq f.deadline$$

Algorithm 1 Fitness Function

```
 1: procedure FITNESS(Genome g)
 2: $makespan \leftarrow 0$
 3: $E_{\text{f.sorted}} \leftarrow$ sort flows based on interdependencies
 4: $\forall f \in E_{f.sorted}$:
 5:     $ST \leftarrow$ f.sender
 6:     $RT \leftarrow$ f.receiver
 7:     $ST.processor \leftarrow p \in ST.CanRunOn$ task's genes
 8:     $RT.processor \leftarrow p \in RT.CanRunOn$ task's genes
 9:     $f.route \leftarrow r \in R$ using flow's genes
10:     $f.InjecTime \leftarrow$ find earliest feasible time slot
11:     $f.arrival \leftarrow f.InjecTime + f.e2eDelay$
12:     if $f.arrival > f.deadline$ then return infinity
13:     $RT.startTime \leftarrow \max(RT.startTime, f.arrival)$
14:     $RT.finishTime \leftarrow RT.startTime + RT.ExTime$
15:     $makespan \leftarrow \max(makespan, RT.finishTime)$
16: return makespan
```

GA Aspects| *Fitness Function -> in Code*

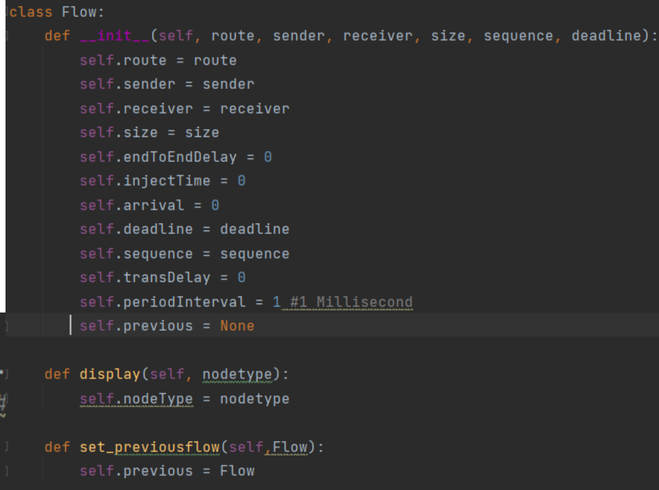


Algorithm 1 Fitness Function

```
1:  procedure FITNESS(Genome g)
2:  $makespan \leftarrow 0$
3:  $E_{\text{f.sorted}} \leftarrow$ sort flows based on interdependencies
4:  $\forall f \in E_{f.sorted}$:
5:      $ST \leftarrow$ f.sender
6:      $RT \leftarrow$ f.receiver
7:      $ST.processor \leftarrow p \in ST.CanRunOn$ task's genes
8:      $RT.processor \leftarrow p \in RT.CanRunOn$ task's genes
9:      $f.route \leftarrow r \in R$ using flow's genes
10:     $f.InjecTime \leftarrow$ find earliest feasible time slot
11:     $f.arrival \leftarrow f.InjecTime + f.e2eDelay$
12:     if $f.arrival > f.deadline$ then return infinity
13:     $RT.startTime \leftarrow \max(RT.startTime, f.arrival)$
14:     $RT.finishTime \leftarrow RT.startTime + RT.ExTime$
15:     $makespan \leftarrow \max(makespan, RT.finishTime)$
16: return makespan
```

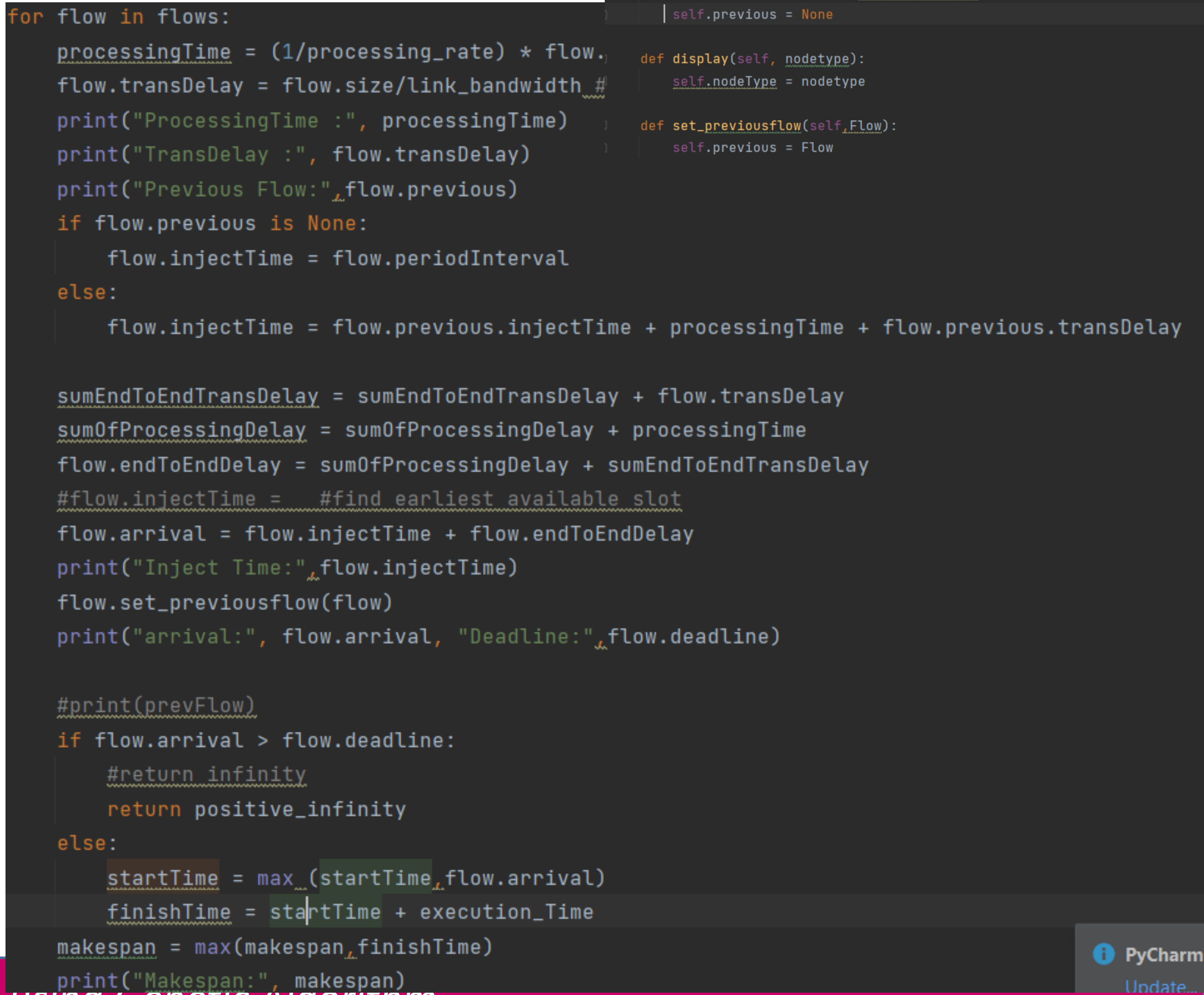
***Note**: Currently static route, inside the sorted flows, f.route also set based on probability matrix*

Flow Class



```python
class Flow:
    def __init__(self, route, sender, receiver, size, sequence, deadline):
        self.route = route
        self.sender = sender
        self.receiver = receiver
        self.size = size
        self.endToEndDelay = 0
        self.injectTime = 0
        self.arrival = 0
        self.deadline = deadline
        self.sequence = sequence
        self.transDelay = 0
        self.periodInterval = 1 #1 Millisecond
        self.previous = None

    def display(self, nodetype):
        self.nodeType = nodetype

    def set_previousflow(self,Flow):
        self.previous = Flow
```

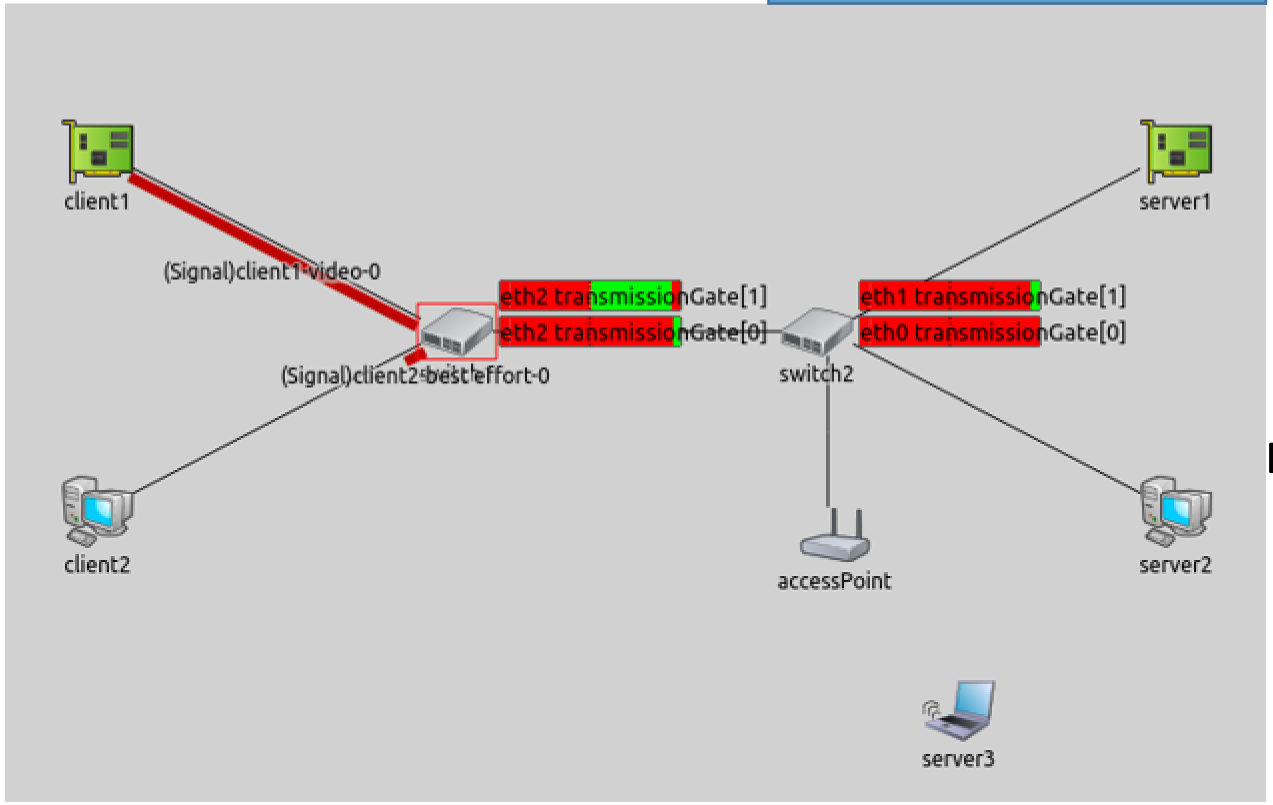
For each flow

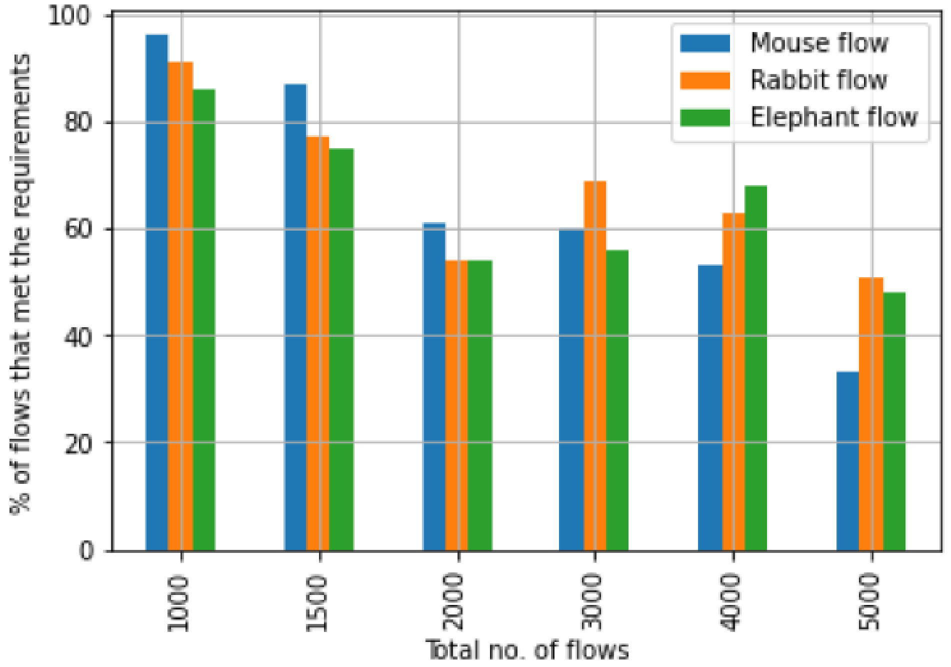


```python
for flow in flows:
    processingTime = (1/processing_rate) * flow.
    flow.transDelay = flow.size/link_bandwidth #
    print("ProcessingTime :", processingTime)
    print("TransDelay :", flow.transDelay)
    print("Previous Flow:",flow.previous)
    if flow.previous is None:
        flow.injectTime = flow.periodInterval
    else:
        flow.injectTime = flow.previous.injectTime + processingTime + flow.previous.transDelay

    sumEndToEndTransDelay = sumEndToEndTransDelay + flow.transDelay
    sumOfProcessingDelay = sumOfProcessingDelay + processingTime
    flow.endToEndDelay = sumOfProcessingDelay + sumEndToEndTransDelay
    #flow.injectTime =   #find earliest available slot
    flow.arrival = flow.injectTime + flow.endToEndDelay
    print("Inject Time:",flow.injectTime)
    flow.set_previousflow(flow)
    print("arrival:", flow.arrival, "Deadline:",flow.deadline)

    #print(prevFlow)
    if flow.arrival > flow.deadline:
        #return infinity
        return positive_infinity
    else:
        startTime = max (startTime,flow.arrival)
        finishTime = startTime + execution_Time
    makespan = max(makespan,finishTime)
    print("Makespan:", makespan)
```

GA Approach| *Deploy in OMNET++ and simulate*

OMNET ++ Simulator





Example Graph (Results)

Fig. 4. Bar plot of % of packets meeting requirements

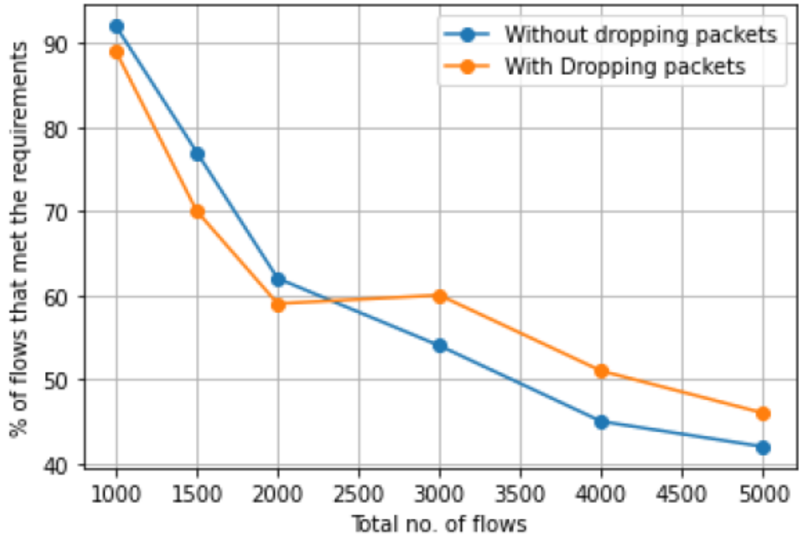


Fig. 5. Comparison of two versions of routing algorithm

Using OMNET++ simulator to run the schedules generated. Results Analysis in terms of Graphs. Demonstrate improvements

Analyze Results