The program used adds a limit on the total number of packets that the destination is able to receive. In order to be tested on only 1 lab machine and 1 raspberry pi, the code has been added into into the calc.p4 file.

To show that the code itself works, after running a certain number of calculations, the switch no longer carries out the calculations. It will proceed to drop the files, reflecting back an error file that gives 3735927486 instead of the correct answer.

If used in forwarding system as a switch with 3 raspberry pis, the second pi (switch) will continue to receive all packets beyond the limit stated (in this case 4) from the sender, but proceed to drop all the remaining packets and not have them forwarded to the destination address (receiver)

C^[[Api@p4pi:~/CWM-ProgNets/assignment6\$ sudo python calc_receiver.py

```
###[ Ethernet ]###
-
###[ Ethernet ]###
                                                                                   = 00:04:00:00:00:00
                                                                        dst
           = 00:04:00:00:00:00
                                                                                   = 00:04:00:00:00:00
           = 00:04:00:00:00:00
                                                                        STC
                                                                                   = 0x1234
           = 0x1234
                                                                        type
###[ P4calc ]###
                                                                      ###[ P4calc ]###
                                                                           Р
    Four
                                                                                         '4'
                                                                           Four
    version
                                                                           version
    ор
                                                                           ор
    operand_a
                                                                           operand_a
    operand_b
                                                                           operand b
###[ Raw ]###
load
                                                                           result
                                                                     ###[ Raw ]###
load
###[ Ethernet ]###
                                                                      10
           = 00:04:00:00:00:00
 SCC
           = 00:04:00:00:00:00
                                                                      ###[ Ethernet ]###
           = 0x1234
                                                                        dst
                                                                                     00:04:00:00:00:00
                                                                        src
                                                                                   = 00:04:00:00:00:00
                                                                                    = 0x1234
    Four
                                                                      ###[ P4calc ]###
     version
                                                                                         'P'
    operand a
                                                                           Four
    operand_b
                                                                           version
                                                                                         0x1
     result
                                                                           OD
###[ Raw ]###
load
                                                                           operand a
                                                                           operand b
                                                                           result
                                                                                         10
                                                                      ###[ Raw ]###
load
                                                                      3735927486
Left photo is the 1<sup>st</sup> 2 calculations, and the photo on the ###[ Ethernet
                                                                                     00:04:00:00:00:00
right is the remaining 2 calculations before getting an
                                                                                     00:04:00:00:00:00
                                                                        src
                                                                                   = 0x1234
error on the 5<sup>th</sup> calculation.
                                                                        type
                                                                      ###[ P4calc ]###
                                                                           Four
                                                                                         141
```

version

op operand_a operand_b result

###[Raw]### load 0x1

3735927486

```
235
       apply {
            if (hdr.p4calc.isValid()) {
236
237
238
             r.read(meta.read from register,0);
              if (meta.read_from_register < 4) {</pre>
239
                    calculate.apply();
240
                    meta.read from register = meta.read from register + 1;
241
                    r.write(0,meta.read from register);
242
243
244
            } else {
245
                operation drop();
246
       }
247
248 }
249
```

The main changes to the p4 code is under the apply section. In order to utilise r.read and r.write, it was necessary to also store it into a variable, in this case meta.read_from_register.

```
2 #define REGISTER_SIZE 1
158 register<br/>
bit<8>>(REGISTER_SIZE) r;
```

The above 2 lines were also added in order to initialise r, and have a register capable of storing it.

Link to use in reality

In reality, this acts as a limiter on the destination side of things. If a timer is added as an additional layer on top of the code, which is very complex, it can allow the register to reset every fix amount of time. This restricts the total number of packets per second that can arrive at the destination side, preventing clients from being flooded by packets in an attack while still optimising the amount of data they can receive.

Limitations are that it does not help telecoms side by reducing wasted packets, which can be improved by implementing a source side limit on number of packets sent to a certain destination. That can help throttle internet usage when there are extremely heavy-duty users.