Федеральное государственное автономное образовательное учреждение высшего образования
«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ»
(РУДН)

Факультет физико-математических и естественных наук,
Кафедра прикладной информатики и теории вероятностей

02.03.01 «Математика и компьютерные науки»

ОТЧЕТ
о прохождении учебной практики (научно-исследовательская работа (получение первичных навыков научно-исследовательской работы))

(вид и наименование практики)

Юхнин Илья Андреевич
(Ф.И.О. обучающегося)

Курс, группа 3, НКНбд-01-19

Место прохождения практики Отдел информационно-технологического обеспечения естественно-научных факультетов УИТО и СТС
полное наименование организации (предприятия)

Сроки прохождения с «18» 04 2022 г. по «19» 06 2022 г.

Руководители практики:

от РУДН к.ф.-м.н., ст.преп. Медведева Е.Г.
Ф.И.О., должность

от организации _____
Ф.И.О., должность

Оценка_____

Москва 2022 г.

Оглавление

Список сокращений

Англоязычные сокращения

PI – Proportional Integral

PIE – Proportional Integral Enhanced

NS – The Network Simulator

RED – Random early detection

WAN - Wide area network

Введение

В рамках учебной практики была выполнена научно-исследовательская работа по теме «Моделирование PI контроллера».

Согласно программе учебной практики направления подготовки «Математика и компьютерные науки» целями практики являются:

- формирование профессиональных навыков в проведении научных исследований;
- формирование навыков использования современных научных методов для решения научных и практических задач;
- формирование практических навыков написания вспомогательных программных комплексов для проведения вычислительных экспериментов;
- формирование общекультурных, общепрофессиональный и профессиональных компетенций в соответствии с ОС ВО РУДН;
- формирование навыков оформления и представления результатов научного исследования;
- формирование навыков работы с источниками данных.

Там же определены задачи практики:

- формирование у студентов навыков в области изучения научной литературы и (или) научно-исследовательских проектов в соответствии с будущим профилем профессиональной деятельности и применения новых научных результатов;
- обучение правильному составлению научных обзоров и отчетов;
- формирование навыков решения конкретных научно-практических задач самостоятельно или в научном коллективе;
- обучение навыкам работы с прикладными комплексами программ для проведения вычислительных экспериментов;
- формирование способности разработки вспомогательных программных инструментов;
- обучение подготовке научных публикаций;
- формирование способности проводить научные исследования и получать новые научные и прикладные результаты.

Описание принципа работы PI контроллера

Обычно пропорционально-интегральный контроллер называют PI-контроллером, его выход состоит из суммы пропорциональных и интегральных управляющих воздействий.
Для настройки регулятора нужно варьировать коэффициенты:

При увеличении kp увеличивается скорость выхода на установленное значение, увеличивается управляющий сигнал. Чисто математически система не может прийти ровно к заданному значению, так как при приближении к установке П составляющая пропорционально уменьшается. При дальнейшем увеличении kp реальная система теряет устойчивость и начинаются колебания.

При увеличении ki растёт скорость компенсации накопившейся ошибки, что позволяет вывести систему точно к заданному значению с течением времени. Если система медленная, а ki слишком большой – интегральная сумма сильно вырастет и произойдёт перерегулирование, которое может иметь характер незатухающих колебаний с большим периодом. Поэтому интегральную сумму в алгоритме регулятора часто ограничивают, чтобы она не могла увеличиваться и уменьшаться до бесконечности.

$$u(t) = P + I = K_p e(t) + K_i \int_o^\tau e(t)dt$$

Рис. 1 Формула PI контроллера

Моделирование PI контроллера в NS-2

Схема модели, для которой будет смоделирована работа PI контроллера (рис. 2)

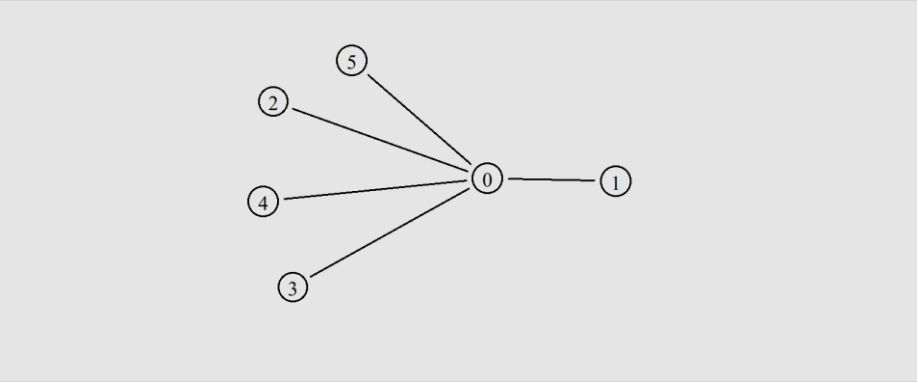

Рис. 2 Топология сети

Узел 1 представляет из себя приемник пакетов, узлы 2,3,4,5 источники пакетов, узел 0 является неким подобием маршрутизатора, использующего очереди DropTail, RED и PI контроллер в разных реализациях.

Характеристика сети: 0-1 имеют пропускную способность 1 Мб и 20 мс с лимитом очереди 100, остальные связи имеют пропускную способность 10 Мб и 1 мс с лимитом 20, размер пакетов 512, используется TCPReno с FTP, время моделирования 100 секунд, используется случайный старт передачи пакетов для разных источников с гарантированным одинаковым временем для всех моделей

Для моделирования и просмотра эффективности контролера написал код для нескольких видов очередей DropTail, RED и самого PI контроллера

Листинг DropTail

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
set tf [open out.tr w]
set windowVsTime [open win w]
set param [open parameters w]
$ns trace-all $tf
proc finish {} {
global ns tf nf
$ns flush-trace
close $tf
close $nf
exec nam out.nam &
exit 0
}
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n2 $n3 1Mb 20ms DropTail
set NumbSrc 4
set Duration 100
for {set j 1} {$j<=$NumbSrc} {incr j} {
set S($j) [$ns node]
}
set rng [new RNG]
$rng seed 3
set RVstart [new RandomVariable/Uniform]
$RVstart set min_ 0
$RVstart set max_ 10
$RVstart use-rng $rng
for {set i 1} {$i<=$NumbSrc} {incr i} {
set startT($i) [expr [$RVstart value]]
set dly($i) 1
puts $param "startT($i) $startT($i) sec"
}
for {set j 1} {$j<=$NumbSrc} {incr j} {
$ns duplex-link $S($j) $n2 10Mb $dly($j)ms DropTail
$ns queue-limit $S($j) $n2 20
}
$ns queue-limit $n2 $n3 100
for {set j 1} {$j<=$NumbSrc} {incr j} {
set tcp_src($j) [new Agent/TCP/Reno]
$tcp_src($j) set window_ 8000
}
```

Листинг RED

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
set tf [open out.tr w]
set windowVsTime [open win w]
set param [open parameters w]
$ns trace-all $tf
proc finish {} {
global ns tf nf
$ns flush-trace
close $tf
close $nf
exec nam out.nam &
exit 0
}
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n2 $n3 1Mb 20ms RED
set NumbSrc 4
set Duration 100
for {set j 1} {$j<=$NumbSrc} {incr j} {
set S($j) [$ns node]
}
set rng [new RNG]
$rng seed 3
set RVstart [new RandomVariable/Uniform]
$RVstart set min_ 0
$RVstart set max_ 10
$RVstart use-rng $rng
for {set i 1} {$i<=$NumbSrc} {incr i} {
set startT($i) [expr [$RVstart value]]
set dly($i) 1
puts $param "startT($i) $startT($i) sec"
}
for {set j 1} {$j<=$NumbSrc} {incr j} {
$ns duplex-link $S($j) $n2 10Mb $dly($j)ms DropTail
$ns queue-limit $S($j) $n2 20
}
$ns queue-limit $n2 $n3 100
for {set j 1} {$j<=$NumbSrc} {incr j} {
set tcp_src($j) [new Agent/TCP/Reno]
$tcp_src($j) set window_ 8000
}
```

Листинг PI контроллер

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
set tf [open out.tr w]
set windowVsTime [open win w]
set param [open parameters w]
$ns trace-all $tf
proc finish {} {
global ns tf nf
$ns flush-trace
close $tf
close $nf
exec nam out.nam &
exit 0
}
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n2 $n3 1Mb 20ms PI
set NumbSrc 4
set Duration 100
for {set j 1} {$j<=$NumbSrc} {incr j} {
set S($j) [$ns node]
}
set rng [new RNG]
$rng seed 3
set RVstart [new RandomVariable/Uniform]
$RVstart set min_ 0
$RVstart set max_ 10
$RVstart use-rng $rng
for {set i 1} {$i<=$NumbSrc} {incr i} {
set startT($i) [expr [$RVstart value]]
set dly($i) 1
puts $param "startT($i) $startT($i) sec"
}
for {set j 1} {$j<=$NumbSrc} {incr j} {
$ns duplex-link $S($j) $n2 10Mb $dly($j)ms DropTail
$ns queue-limit $S($j) $n2 20
}
$ns queue-limit $n2 $n3 100
for {set j 1} {$j<=$NumbSrc} {incr j} {
set tcp_src($j) [new Agent/TCP/Reno]
$tcp_src($j) set window_ 8000
}
```

```
for {set j 1} {$j<=$NumbSrc} {incr j} {
set tcp_snk($j) [new Agent/TCPSink]
}
for {set j 1} {$j<=$NumbSrc} {incr j} {
$ns attach-agent $S($j) $tcp_src($j)
$ns attach-agent $n3 $tcp_snk($j)
$ns connect $tcp_src($j) $tcp_snk($j)
}
for {set j 1} {$j<=$NumbSrc} {incr j} {
set ftp($j) [$tcp_src($j) attach-source FTP]
}
for {set j 1} {$j<=$NumbSrc} {incr j} {
$tcp_src($j) set packetSize_ 512
}
for {set i 1} {$i<=$NumbSrc} {incr i} {
$ns at $startT($i) "$ftp($i) start"
$ns at $Duration "$ftp($i) stop"
}
proc plotWindow {tcpSource file k} {
global ns NumbSrc
set time 0.03
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
if {$k == 1} {
puts -nonewline $file "$now \t $cwnd \t"
} else {
if {$k < $NumbSrc} {
puts -nonewline $file "$cwnd \t" }
}
if {$k == $NumbSrc} {
puts -nonewline $file "$cwnd \n"}
$ns at [expr $now+$time] "plotWindow $tcpSource $file $k"}
for {set j 1} {$j<=$NumbSrc} {incr j} {
$ns at 0.1 "plotWindow $tcp_src($j) $windowVsTime $j"
}
set qfile [$ns monitor-queue $n2 $n3 [open queue.tr w] 0.05]
[$ns link $n2 $n3] queue-sample-timeout;
$ns at [expr $Duration] "finish"
$ns run
```

```
for {set j 1} {$j<=$NumbSrc} {incr j} {
set tcp_snk($j) [new Agent/TCPSink]
}
for {set j 1} {$j<=$NumbSrc} {incr j} {
$ns attach-agent $S($j) $tcp_src($j)
$ns attach-agent $n3 $tcp_snk($j)
$ns connect $tcp_src($j) $tcp_snk($j)
}
for {set j 1} {$j<=$NumbSrc} {incr j} {
set ftp($j) [$tcp_src($j) attach-source FTP]
}
for {set j 1} {$j<=$NumbSrc} {incr j} {
$tcp_src($j) set packetSize_ 512
}
for {set i 1} {$i<=$NumbSrc} {incr i} {
$ns at $startT($i) "$ftp($i) start"
$ns at $Duration "$ftp($i) stop"
}
proc plotWindow {tcpSource file k} {
global ns NumbSrc
set time 0.03
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
if {$k == 1} {
puts -nonewline $file "$now \t $cwnd \t"
} else {
if {$k < $NumbSrc} {
puts -nonewline $file "$cwnd \t" }
}
if {$k == $NumbSrc} {
puts -nonewline $file "$cwnd \n"}
$ns at [expr $now+$time] "plotWindow $tcpSource $file $k"}

for {set j 1} {$j<=$NumbSrc} {incr j} {
$ns at 0.1 "plotWindow $tcp_src($j) $windowVsTime $j"
}
set qfile [$ns monitor-queue $n2 $n3 [open queue.tr w] 0.05]
[$ns link $n2 $n3] queue-sample-timeout;
$ns at [expr $Duration] "finish"
$ns run
```

```
for {set j 1} {$j<=$NumbSrc} {incr j} {
set tcp_snk($j) [new Agent/TCPSink]
}
for {set j 1} {$j<=$NumbSrc} {incr j} {
$ns attach-agent $S($j) $tcp_src($j)
$ns attach-agent $n3 $tcp_snk($j)
$ns connect $tcp_src($j) $tcp_snk($j)
}
for {set j 1} {$j<=$NumbSrc} {incr j} {
set ftp($j) [$tcp_src($j) attach-source FTP]
}
for {set j 1} {$j<=$NumbSrc} {incr j} {
$tcp_src($j) set packetSize_ 512
}
for {set i 1} {$i<=$NumbSrc} {incr i} {
$ns at $startT($i) "$ftp($i) start"
$ns at $Duration "$ftp($i) stop"
}
proc plotWindow {tcpSource file k} {
global ns NumbSrc
set time 0.03
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
if {$k == 1} {
puts -nonewline $file "$now \t $cwnd \t"
} else {
if {$k < $NumbSrc} {
puts -nonewline $file "$cwnd \t" }
}
if {$k == $NumbSrc} {
puts -nonewline $file "$cwnd \n"}
$ns at [expr $now+$time] "plotWindow $tcpSource $file $k"}
for {set j 1} {$j<=$NumbSrc} {incr j} {
$ns at 0.1 "plotWindow $tcp_src($j) $windowVsTime $j"
}
set qfile [$ns monitor-queue $n2 $n3 [open queue.tr w] 0.05]
[$ns link $n2 $n3] queue-sample-timeout;
$ns at [expr $Duration] "finish"
$ns run
```

Далее с помощью GNUplot создал графики очереди и размера окна для узлов 0-1

Листинг GNUplot очередь

```
#!/usr/bin/gnuplot -persist
set encoding utf8
set term pdfcairo font "Arial,9"
set out 'queue.pdf'
set style line 2
set xlabel "time"
plot "queue.tr" using ($1):($5) with lines title "Queue"
```

Листинг GNUplot размер окна

```
#!/usr/bin/gnuplot -persist
set encoding utf8
set term pdfcairo font "Arial,9"
set out 'win.pdf'
set style line 2
set xlabel "time"
set ylabel "size"
plot "win" using ($1):($2) with lines title "Size"
```

В итоге получились данные графики

DropTail

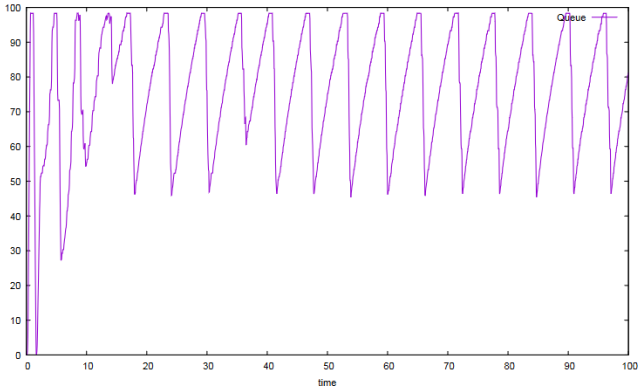

Рис. 3 График очереди DropTail

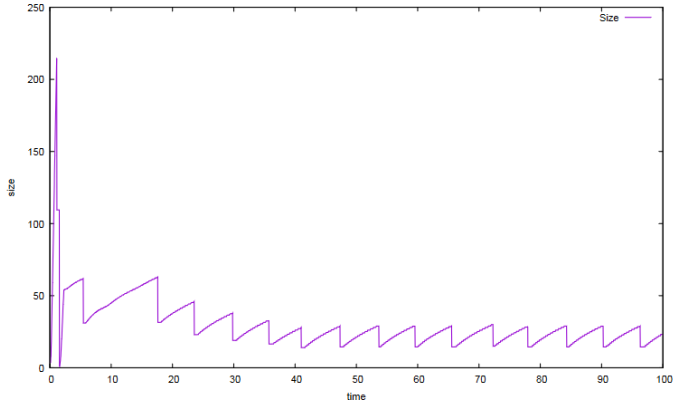

Рис. 4 График размера окна DropTail

RED

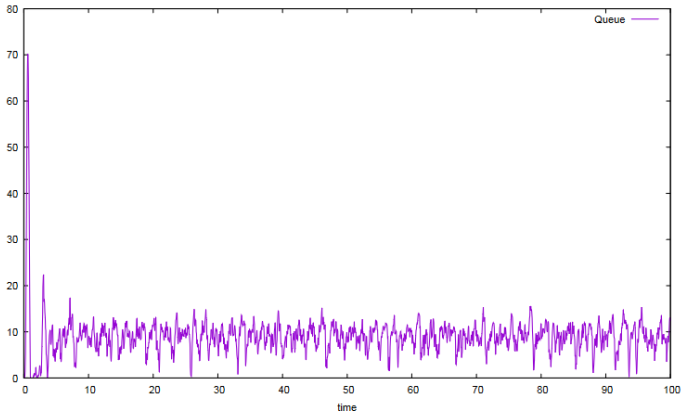

Рис. 5 График очереди RED

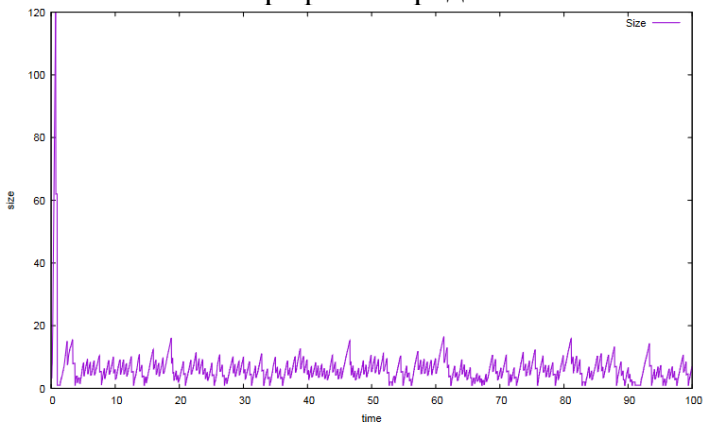

Рис. 6 График размера окна RED

PI контроллер

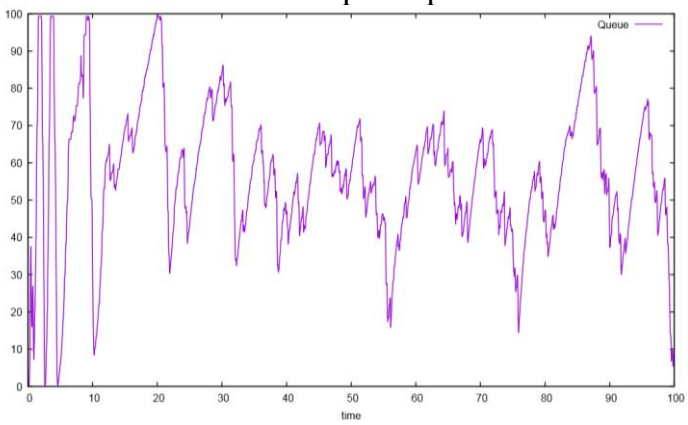

Рис. 7 График очереди PI контроллер

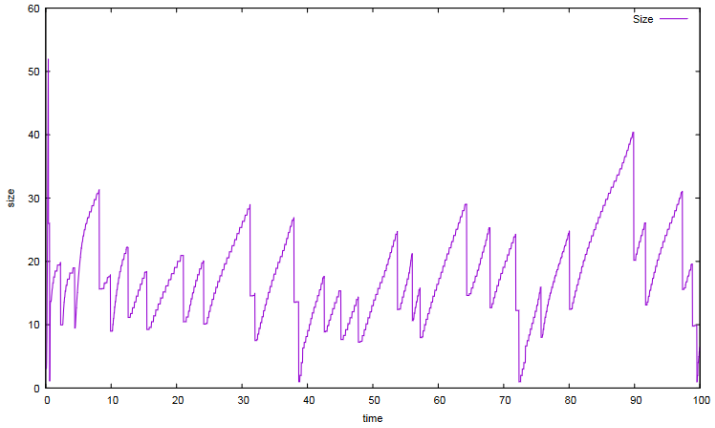

Рис. 8 График размера окна PI контроллер

DropTail представляет из себя простой механизм сброса пакетов при переполнении и как видно по графикам он постоянно переполняется и сбрасывает пакеты.

Исходя из полученных данных видно, что RED справляется со задачей контроля очереди, среди выбранных, лучше всего, не допуская переполнения очереди для данной схемы с последующей потерей пакетов.

PI контроллер имеет лучше результаты по сравнению с DropTail и имеет несколько переполнений в начале симуляции, но в последующем он не переполняется.

Также данные модели работают в Network Animator, ниже представлены несколько моментов DropTail в момент сброса пакетов (рис.9).

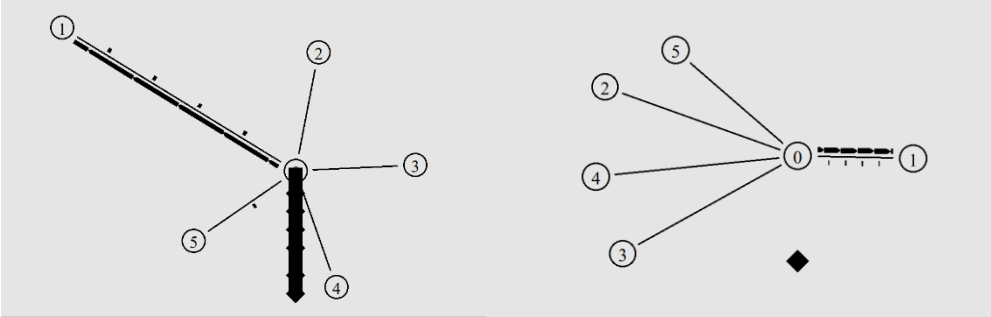

Рис. 9 Network Animator

Моделирование PI контроллера в NS-3

NS-3 имеет более сложную структуру создания сети и добавление очереди, также все программы в нем используют только язык программирования C++ из-за этого совместимость с NS-2 только для тех программ, которые написаны на C++, а не на OTcl, поэтому смоделирую готовый пример PI в данном случае встроенный в NS-3 pie-example.cc.
Схема сети представлена на рисунке (рис. 10), соединение между узлами 2-3 имеет PIE очередь.
Время моделирования 10 сек, данный пример проверяет размер очереди каждые 0.01 секунды, использует TCPNewReno, падений не было, так как очередь не переполнялась. Чтобы увидеть работу PIE контроллера, отключил его и получил результаты длины работы только DropTail и перед этим PIE, сравнивая оба результата (рис. 11-12) видно, что PIE работает и контролирует очередь, уменьшая нагрузку.

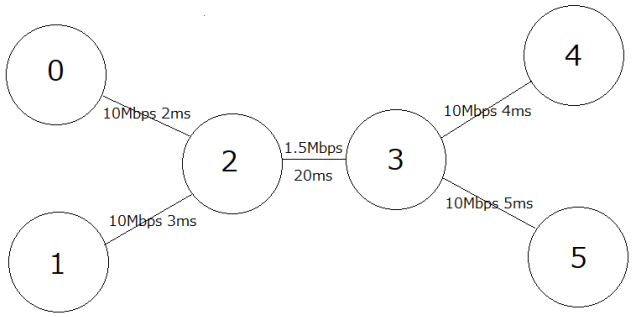

Рис. 10 Схема сети

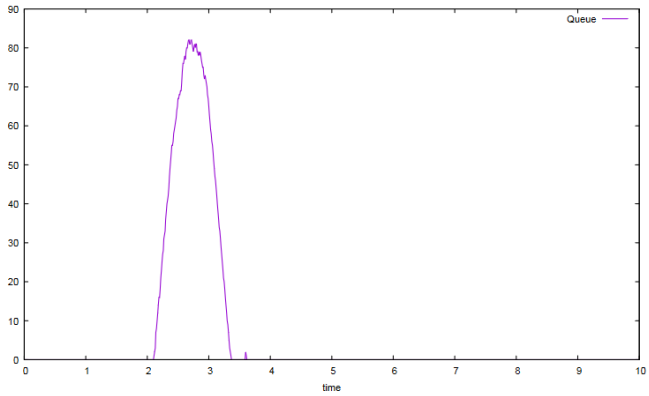

Рис. 11 Длина очереди PIE

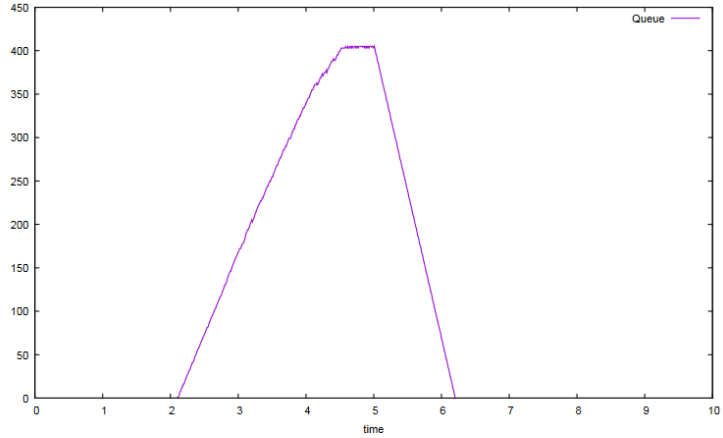

Рис. 12 Длина очереди DropTail

Моделирование PI контроллера в Mininet

PI в сетях и Mininet состоит из трех простых базовых компонентов: 1) случайное отбрасывание при постановке в очередь; 2) периодическое обновление информации о вероятности сброса; 3) расчет задержки. Когда приходит пакет, принимается случайное решение о том, следует ли его отбросить. Вероятность отбрасывания периодически обновляется в зависимости от того, насколько текущая задержка отличается от целевого значения, а также в зависимости от того, увеличивается или уменьшается задержка в очереди. Задержка в очереди может быть получена с помощью прямых измерений или с использованием оценок, рассчитанных на основе длины очереди и скорости удаления из очереди.
Начну моделирование с создания простой топологии сети в Mininet с помощью MiniEdit (рис. 13). Топология использует 10.0.0.0/8 которые представляют собой стандартную сеть в Mininet.

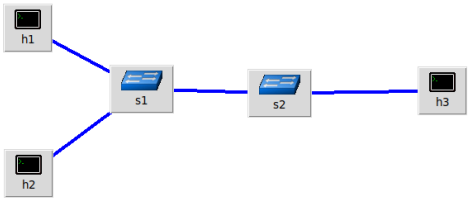

Рис. 13 Сеть в Miniedit

Я решил эмулировать WAN сеть с высокой задержкой и задал задержку 40 мс на интерфейсе s1-eth2 коммутатора s1 с помощью команды:

sudo tc qdisc add dev s1-eth1 root netem delay 40ms

Далее пропинговал h3 с h1 и h2 чтобы проверить работу сети (рис. 14)

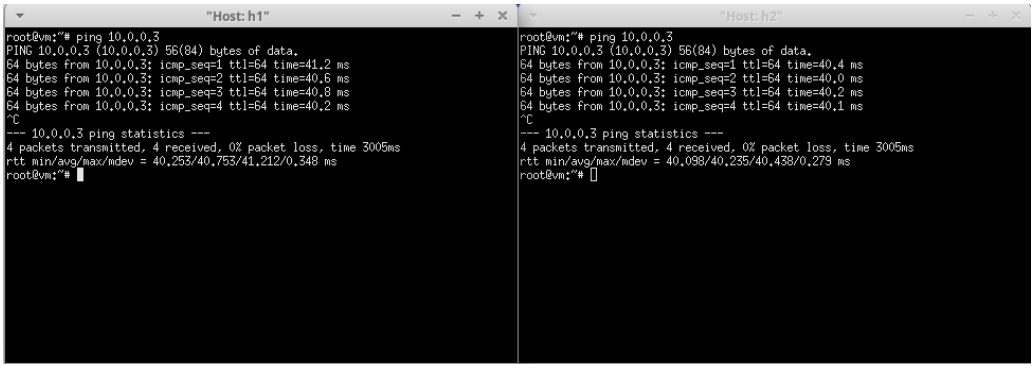

Рис. 14 Терминалы h1, h2

Потерь пакетов нет, задержка в пределах 40 мс, сеть работает.

Задаю правило ограничения скорости на интерфейсе s2-eth2 коммутатора s2:

sudo tc qdisc add dev s2-eth2 root handle 1: tbf rate 1gbit burst 500000 limit 25000000

limit – количество байтов, которые могут быть поставлены в очередь ожидания чтобы токены стали доступны.

burst – это максимальное количество байтов, для которого токены могут быть доступны мгновенно.

rate – скорость передачи.

И далее задаю параметры для PIE представляющую собой PI контроллер в Mininet:

sudo tc qdisc add dev s2-eth2 parent 1: handle 2: pie limit 15000 target 45ms tupdate 10ms alpha 5 beta 30

limit – размер пакета при превышении которого происходит сброс пакетов.

target – ожидаемая задержка очереди.

tupdate – частота с которой происходит вычисление возможности сброса пакетов.

alpha, beta – параметры для контроля возможности сброса пакетов.

После задания параметров сети запустил iPerf3 в режиме сервера на терминале хоста h3, который получал от хоста h1 трафик в течение 100 секунд, в результате было отправлено 10.9 Гб данных, а средняя скорость равна 940 Мбит/с, что примерно равно заданной скорости канала в 1 Гбит/с (рис. 15). Также вывел график окна перегрузки хоста h1 с пиковым значением в районе 15000 байт.

```
[ 17]  95.00-96.00  sec   113 MBytes   950 Mbits/sec
[ 17]  96.00-97.00  sec   112 MBytes   936 Mbits/sec
[ 17]  97.00-98.00  sec   113 MBytes   949 Mbits/sec
[ 17]  98.00-99.00  sec   113 MBytes   944 Mbits/sec
[ 17]  99.00-100.00 sec   114 MBytes   955 Mbits/sec
[ 17] 100.00-100.06 sec  6.71 MBytes   939 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[ 17]   0.00-100.06 sec  0.00 Bytes  0.00 bits/sec                  sender
[ 17]   0.00-100.06 sec  10.9 GBytes   940 Mbits/sec                  receiver
-----------------------------------------------------------
```

Рис. 15 Вывод терминала h3

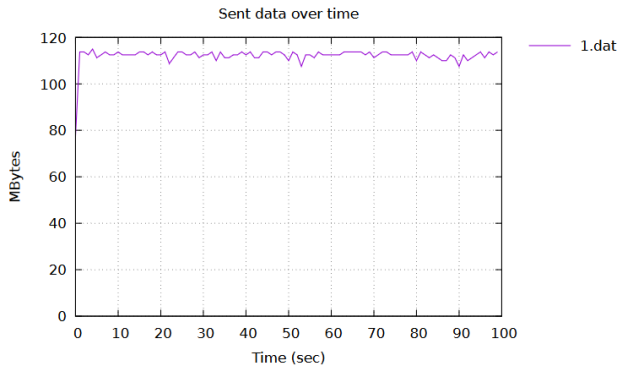

Рис. 16 График скорости передачи

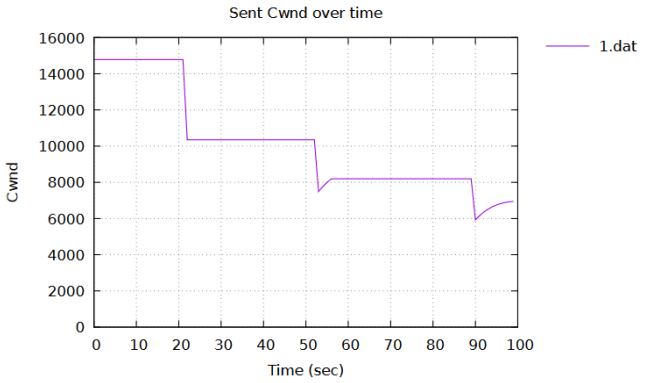

Рис. 17 График окна перегрузки хоста h1

Также с помощью хоста h2 рассмотрел время приема передачи с помощью команды ping. Приведенный ниже результат (рис. 18) показывает, что все 100 пакетов были получены успешно, и что минимальное, среднее, максимальное и стандартное отклонение равны 49.073, 67.971, 75.495, 4.722 мс соответственно. Выходные данные также подтверждают отсутствие переполнения буфера, поскольку средняя задержка (67.971 мс) незначительно превышает настроенную задержку (40 мс).

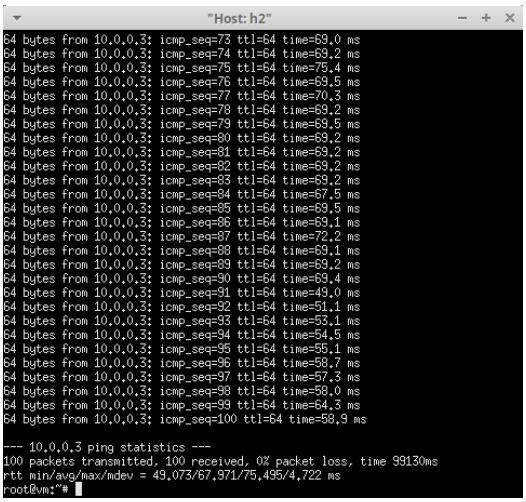

Рис. 18 Вывод терминала h2

Так как уже известно, что PI контроллер зависит от длины очереди и задержки, то я решил проверить зависит ли PI контроллер от состояния сети и изменил его, понизив скорость пропускного канала до 100 Мбит/с. В результате (рис. 19) было отправлено 1.11 Гб данных, а средняя скорость равна 95.4 Мбит/с, что примерно равно заданной скорости канала в 100 Мбит/с. Пиковое значение окна перегрузки хоста h1 в районе 3000 байт.

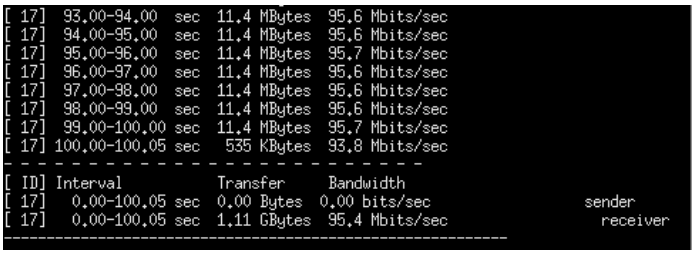

Рис. 19 Вывод терминала h3 100 Мбит/с

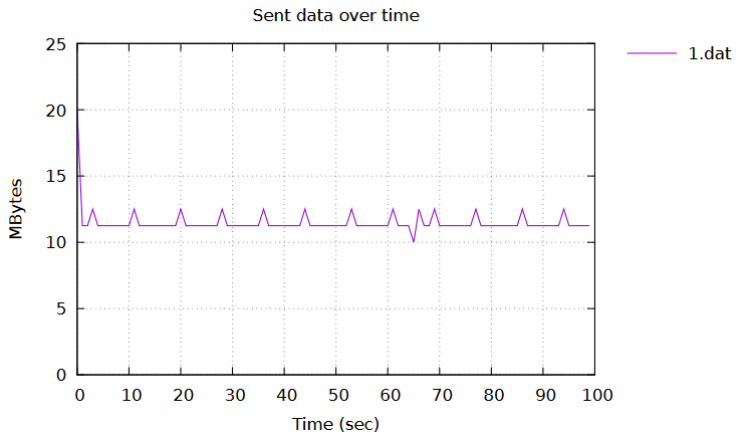

Рис. 20 График скорости передачи 100 Мбит/с

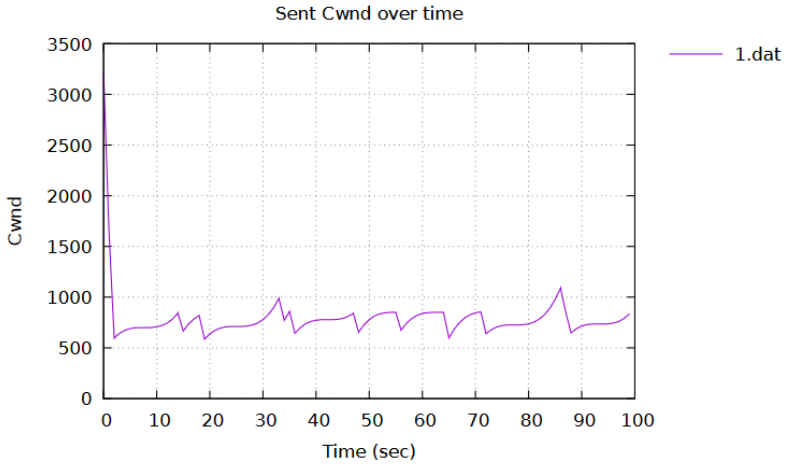

Рис. 21 График окна перегрузки хоста h1

Все 100 пакетов были получены успешно (рис. 22), минимальное, среднее, максимальное и стандартное отклонение равны 40.431, 58.971, 94.867, 11.077 мс соответственно. Выходные данные подтверждают отсутствие переполнения буфера, поскольку средняя задержка (58.971 мс) незначительно превышает настроенную задержку (40 мс).

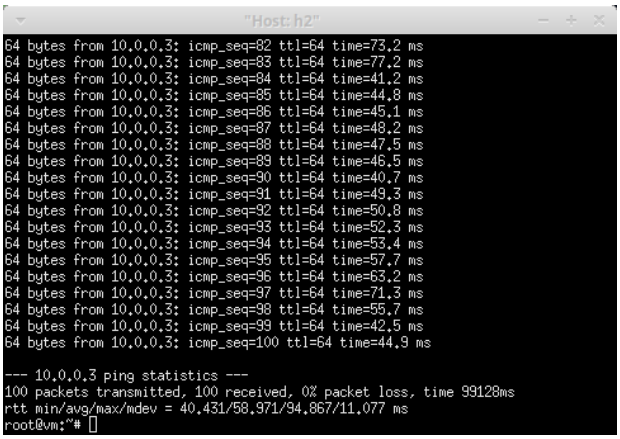

Рис. 22 Вывод терминала h2 100 Мбит/с

В результате полученных данных состояние сети не влияет на PI контроллер.

Реализация PI контроллера в Julia

Реализовал PI контроллер с помощью языка программирования Julia
Листинг:

```
using ModelingToolkit, DifferentialEquations, Plots

@variables t y(t) e(t) de(t) x(t) dx(t) ddx(t)
D = Differential(t)
@parameters kp ki

sp(t) = 100

@named sys = ODESystem([
    D(y) ~ kp*de + ki*e
    D(e) ~ de
    D(x) ~ dx
    D(dx) ~ ddx
    0.0 ~ (y)  - (ddx + dx  + x)
    0.0 ~ (e) - (sp(t) - x)
        ])
p = [1,1]
u0 = zeros(6)
u0[3] = 0
time = (0.0,50.0)
solution = solve(ODEProblem(sys, u0, time, p),ImplicitEuler(),adaptive=false, dt=4e-4)

u = hcat(solution.u...)

plot(solution.t, u[3,:])
```

Создал несколько графиков с разными параметрами kp, ki, начальным положением, временем, точкой назначения:
1. Изначальный график с параметрами по умолчанию (как в листинге)
kp = 1, ki = 1, x0 = 0, time = 50, sp = 100

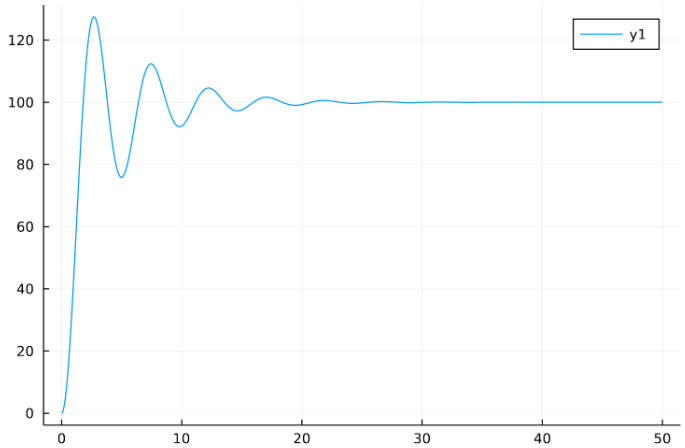

Рис. 23 График 1

2. kp = 10, ki = 2, x0 = 100, time = 70, sp = 2000

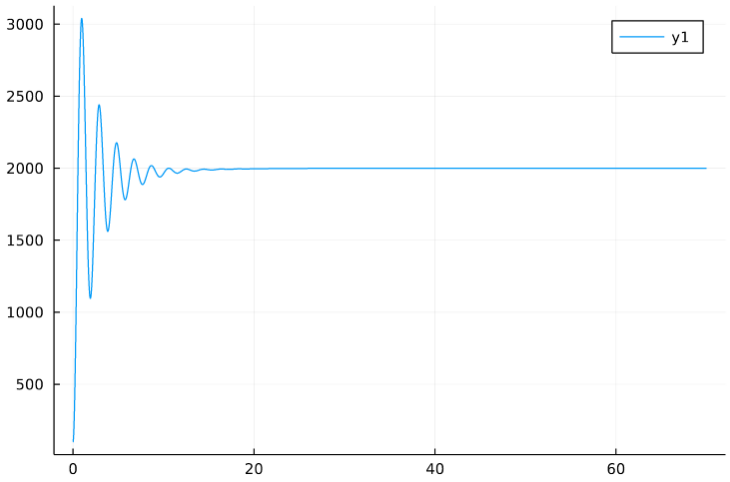

Рис. 24 График 2

3. kp = 1, ki = 0.5, x0 = 0, time = 50, sp = 20

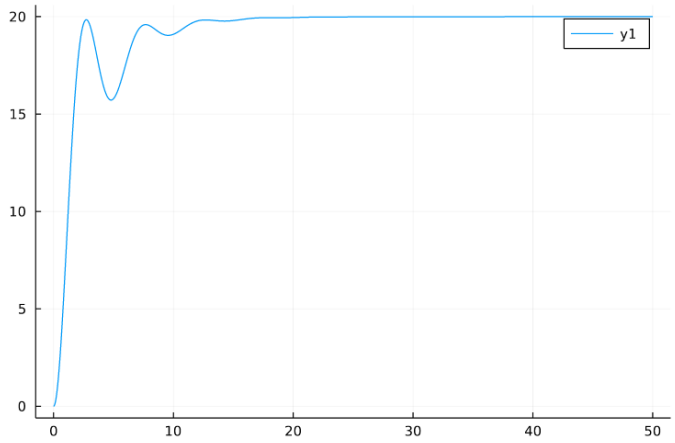

Рис. 25 График 3

4. kp = 3, ki = 0.2, x0 = 0, time = 50, sp = 100

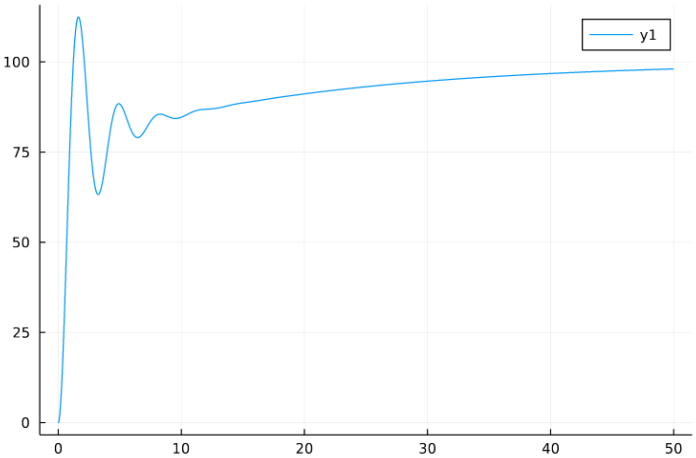

Рис. 26 График 4

Заключение

В ходе научно-исследовательской работы я изучил и смоделировал различные виды очереди в NS-2 (DropTail, RED, PI контроллер), смоделировал работу очередей (PIE, DropTail) в NS-3, смоделировал сеть с PI контроллером в Mininet, изучил язык программирования Julia и с его помощью создал PI контроллер.

Так же я получил следующие навыки:

- формирование профессиональных навыков в проведении научных исследований;
- формирование навыков использования современных научных методов для решения научных и практических задач;
- формирование практических навыков написания вспомогательных программных комплексов для проведения вычислительных экспериментов;
- формирование общекультурных, общепрофессиональный и профессиональных компетенций в соответствии с ОС ВО РУДН;
- формирование навыков оформления и представления результатов научного исследования;
- формирование навыков работы с источниками данных.

Список источников

1. Поляков К. Ю. Основы теории автоматического управления: учеб. пособие. — СПб.: Изд-во СПбГМТУ, 2012. — 234 с.
2. The Network Simulator - ns-2
–URL: https://www.isi.edu/nsnam/ns/
[Дата обращения: 25.04.2022].
3. Network Simulator – ns-3
-URL: https://www.nsnam.org/documentation/
[Дата обращения: 05.05.2022].
4. Mininet Documentation
–URL: https://github.com/mininet/mininet/wiki/Documentation
[Дата обращения: 12.05.2022].
5. Julia 1.7 Documentation
–URL: https://docs.julialang.org/en/v1/
[Дата обращения: 18.05.2022].