МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И КОМПЬЮТЕРНЫХ СЕТЕЙ

УДК 004.72

МОДЕЛИРОВАНИЕ ПРОЦЕССА ПЕРЕДАЧИ ТРАФИКА В ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЯХ

К. И. Никишин, к.т.н., старший преподаватель кафедры ВТ ПГУ, Пенза, Россия;
orcid.org/0000-0001-7966-7833, e-mail: nkipnz@mail.ru

Управление трафиком, критериями в программно-конфигурируемых сетях (ПКС) выполняется контроллером в связке с коммутатором, который работает по открытому протоколу OpenFlow. Протокол OpenFlow является основной частью ПКС. Таким образом, открытый протокол OpenFlow успешно позволяет справляться с возросшими требованиями к пользовательскому трафику, приоритету. ***Цель работы*** *– исследование и моделирование передачи гетерогенного разнородного пользовательского трафика в коммутаторе OpenFlow ПКС на основе цветных временных иерархических сетей Петри и использование пакета моделирования CPN Tools, а также оценка времени потери трафика в случае не найденного правила в таблицах потоков. Задачами исследования являются разработка и построение подсетей чтения заданного трафика, сравнение с правилами кадров в таблицах потоков, классификация трафика и обработка последующих действий с входным трафиком в коммутаторе. Иерархическая модель на сетях Петри позволила исследовать не просто функционирование и поведение ПКС, но и долю потерянных кадров реального времени в сети, а также задержку.*

Ключевые слова: *программно-конфигурируемые сети, контроллер, коммутатор, Ethernet, OpenFlow, таблицы потоков, тайм-ауты, сети Петри, CPN Tools.*

DOI: 10.21667/1995-4565-2022-81-32-41

Введение

При передаче разнородного гетерогенного пользовательского трафика важными критериями в сети являются задержка кадра, быстродействие, полоса пропускания в коммутаторе Ethernet [1, 2], переполнение очередей в коммутаторе. Многие эти критерии становятся управляемыми в программно-конфигурируемых сетях, в отличие от традиционных компьютерных сетей.

Управление трафиком, критериями в ПКС выполняется контроллером в связке с коммутатором, который работает по открытому протоколу OpenFlow [3]. Протокол OpenFlow является основной частью ПКС. Таким образом, открытый протокол OpenFlow успешно позволяет справляться с возросшими требованиями к пользовательскому трафику, приоритету.

Теоретическая часть

Под разнородным гетерогенным пользовательским трафиком понимается сочетание различных видов трафика: трафика реального времени, для которого основные требования – минимальная задержка кадра в сети и максимальное быстродействие, и стандартный трафик, к которому предъявляются гораздо меньшие требования, чем к трафику реального времени.

К таким видам трафика можно отнести служебный трафик, который передает управляющие команды, команды синхронизации в сети, голос и видео, относящиеся к трафику реаль-

ного времени, фоновый трафик, обычный трафик в LAN сети и другие. Каждый вид трафика должен обладать приоритетом в формате кадра Ethernet согласно стандарту IEEE 801.2 [4].

Открытый стандарт OpenFlow позволяет обрабатывать разный вид трафика и управлять им [5]. Схематично обработку трафика в ПКС можно описать следующим образом: поступающий входной трафик перенаправляется на вход одного из портов коммутатора OpenFlow. Трафик поступает с уровня инфраструктуры на уровень управления.

В коммутаторе OpenFlow и контроллере ПКС можно выделить следующие блоки: блок сбора статистической информации, блок классификации и блок принятия решения [6]. Блок сбора статистической информации содержит основные метрики ПКС.

Блок классификации осуществляет передачу трафика в зависимости от его приоритета. Блок принятия решения выполняет определенное действие с пришедшим кадром в зависимости от приоритета кадра, метрик ПКС, задержки по определенному маршруту и т.д. Данные блоки представлены на рисунке 1.

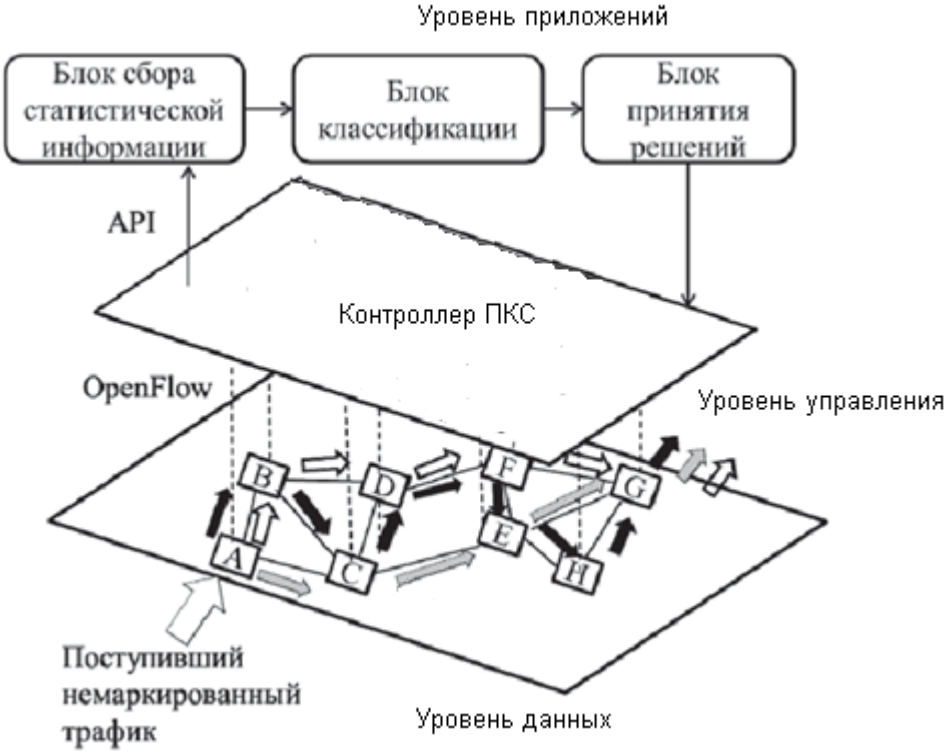

Рисунок 1 – Передача трафика в ПКС
Figure 1 – Traffic transmission in SDN

В экспериментальной части будет рассмотрена передача разнородного гетерогенного пользовательского трафика на основе цветных временных иерархических сетей Петри с помощью свободного распространяемого пакета CPN Tools, который наилучшим образом подходит для исследования компьютерных сетей, их критериев, метрик и задержки в сети [2, 7].

Экспериментальные исследования

Верхний уровень коммутатора OpenFlow представлен на рисунке 2, и можно выделить следующие блоки: подсеть Input traffic SDN, через которую поступает трафик ПКС, непосредственно коммутатор OpenFlow, работающий по данному протоколу, выполняет сравнение маскированного трафика с таблицами потоков, классификацию и дальнейшую передачу трафика при удовлетворении того или иного условия, и подсеть OutPort N, в которую передается трафик с учетом временных параметров и задержки в сети.

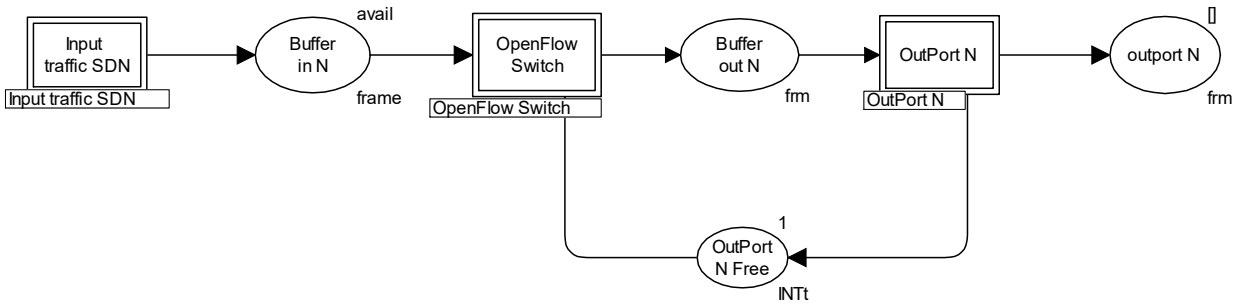

Рисунок 2 – Верхний уровень коммутатора OpenFlow
Figure 2 – Hierarchical level of OpenFlow switch

Подсеть Input traffic SDN представляет собой чтение сгенерированного трафика из файла, соответствующего стандарту OpenFlow с необходимым перечнем полей, и представлена на рисунке 3. Прочтение файла реализуется на переходе Set SDN Frame param с помощью языка CPN ML. При передаче трафика участвуют в вычислениях задержки @+(delay-delta). Кадр передается в следующем виде – 1`f(number,inport,src_MAC,dst_MAC, vlid,qos,src_IP, dst_IP, szfrm,delay,delay2). Более подробно о генераторе трафика, его прочтении в компьютерных сетях написано в статье [8].

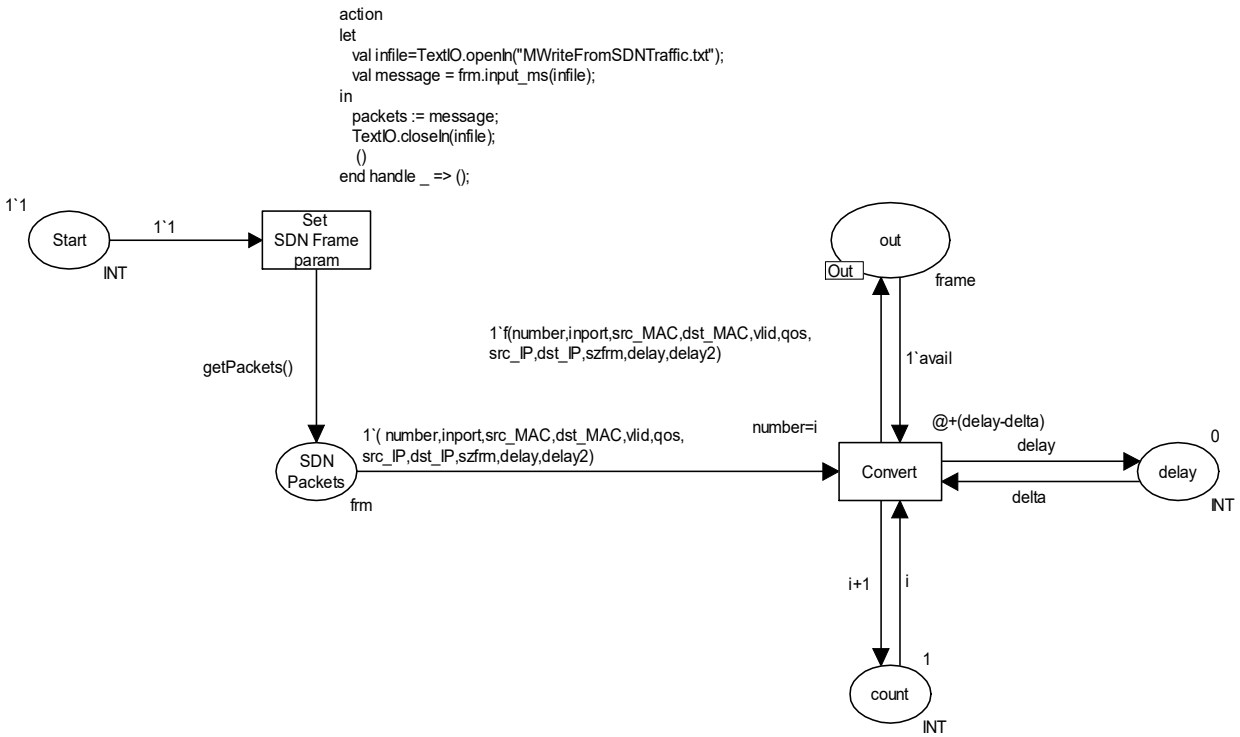

Рисунок 3 – Подсеть чтения трафика в ПКС
Figure 3 – Subnet of read traffic in SDN

Подсеть коммутатора OpenFlow представляет (рисунок 4) собой позицию Buffer In N, через которую поступает входной трафик, подсеть Flow Table1, которая отвечает за таблицы потоков, классификацию трафика и принятие решения по этому трафику, подсеть очередей Queue1 и Queue2, подсеть диспетчера Dispatcher и выходную позицию коммутатора Buffer out N [9].

Очередь 1 предназначена для передачи трафика реального времени с максимальным приоритетом (поле качества обслуживания qos = 7), а очередь 2 – для передачи другого стандартного вида трафика. Подсеть диспетчера трафика выполняет управление очередями, в коммутаторах могут использоваться различные виды диспетчеризации: циклические, дефицитные, справедливые и другие. Моделирование алгоритмов диспетчеров описано в стать-

ях [1, 2]. В модели не исследуются различные виды диспетчеризации, поэтому рассматривается простой тип FIFO по управлению очередями.

Поступающий кадр передается в таблицы потоков, и выполняется сравнение с правилами таблицы потоков 1 через подсеть Matching1 (показана на рисунке 5). На вход перехода Matching Flow Table 1 поступают кадр и таблица потоков 1. В позиции Drop Frames 1 происходит удаление кадров в случае, если не выполняются действия с тайм-аутами (жесткий тайм-аут hard_timeout и таймаут передачи кадра idle_timeout). Если переменная delay из кадра будет больше hard_timeout, то кадр удаляется. Если сумма переменной delay и размер передачи кадра во времени szfrm будут больше idle_timeout, то кадр также удаляется. Данный алгоритм описывается следующим образом: if delay > hard_timeout then 1`(number,inport,src_MAC,dst_MAC,vlid,qos, src_IP, dst_IP,szfrm,delay,0) else if (delay+szfrm) > idle_timeout then 1`(number,inport,src_MAC, dst_MAC,vlid,qos, src_IP,dst_IP,szfrm,delay,0) else empty.

В случае невыявления правила из таблицы потоков 1 кадр передается в позицию Frame to Next Flow Table для сравнения с таблицей потоков N.

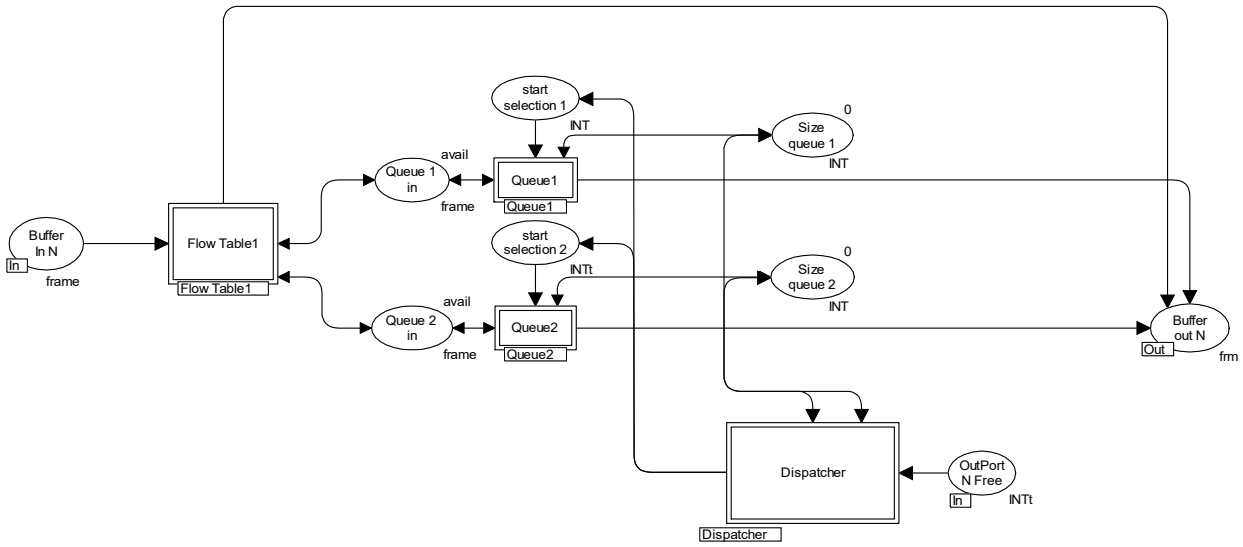

Рисунок 4 – Подсеть коммутатора OpenFlow
Figure 4 – Subnet of OpenFlow switch

Сравнение происходит по определенным немаскируемым полям кадра: MAC адреса приемника или источника, IP адрес источника или входящий порт по таблице и описано следующим образом: if (src_MAC<>src_MAC1) orelse (dst_MAC<>dst_MAC1) orelse (dst_IP<>dst_IP1) orelse (inport<>inport1) then f(number,inport,src_MAC,dst_MAC, vlid,qos, src_IP,dst_IP,szfrm,delay,0) else avail.

В случае нахождения правила по кадру в таблице потоков 1 происходит передача кадра в подсеть классификации Classifier. Подсеть Matching N таблицы потоков N отличается от подсети таблицы потоков 1 тем, что добавляется условие в случае, когда правило для кадра не находится и в этой таблице, то такой кадр тоже помещается в позицию удаленных кадров Drop Frames N. Подсеть Matching N представлена на рисунке 6. Полное условие срабатывания для удаления кадров представлено ниже:

if delay > hard_timeout then 1`(number,inport,src_MAC,dst_MAC,vlid,qos,src_IP, dst_IP, szfrm, delay,0) else if (delay+szfrm) > idle_timeout then 1`(number,inport,src_MAC, dst_MAC, vlid,qos, src_IP,dst_IP,szfrm,delay,0)

else if (src_MAC<>src_MAC1) orelse (dst_MAC<>dst_MAC1) orelse (dst_IP<>dst_IP1) orelse (inport<>inport1) then 1`(number,inport,src_MAC,dst_MAC,vlid,qos, src_IP,dst_IP,szfrm, delay,0)

else empty

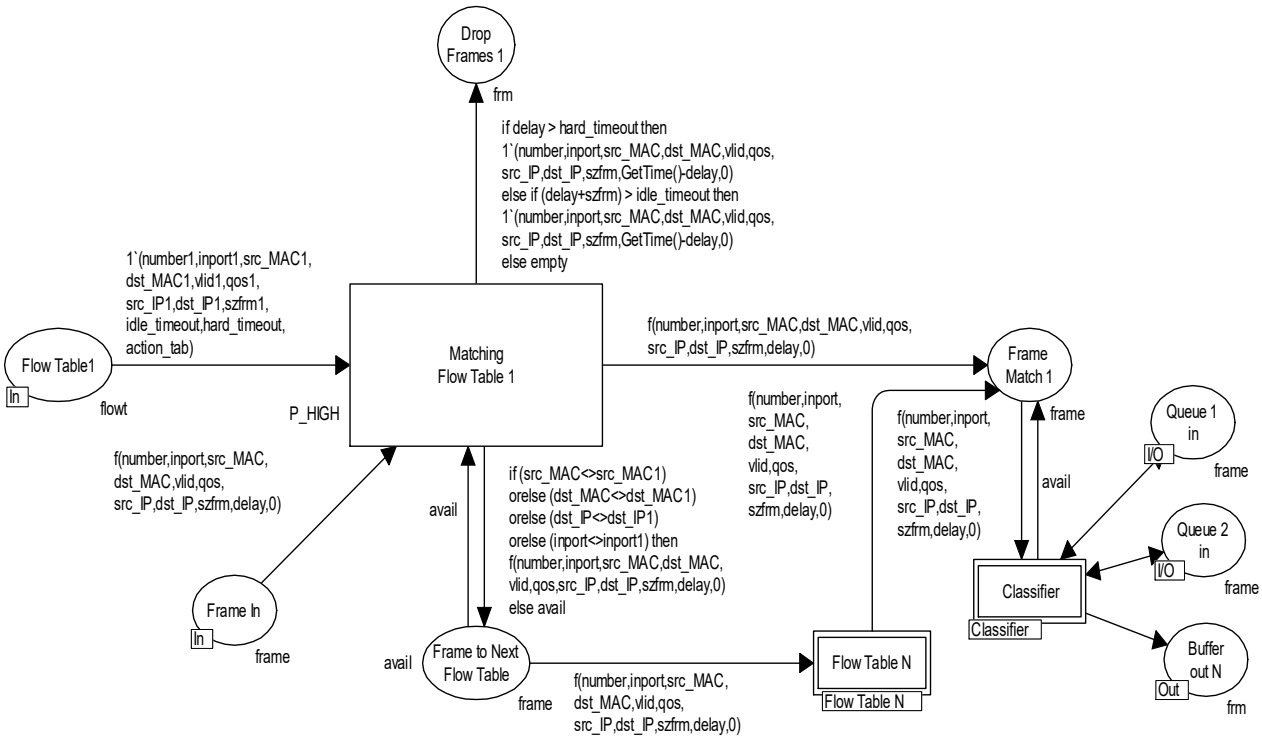

Рисунок 5 – Подсеть сравнения таблицы потоков 1
Figure 5 – Subnet of matching of flow table 1

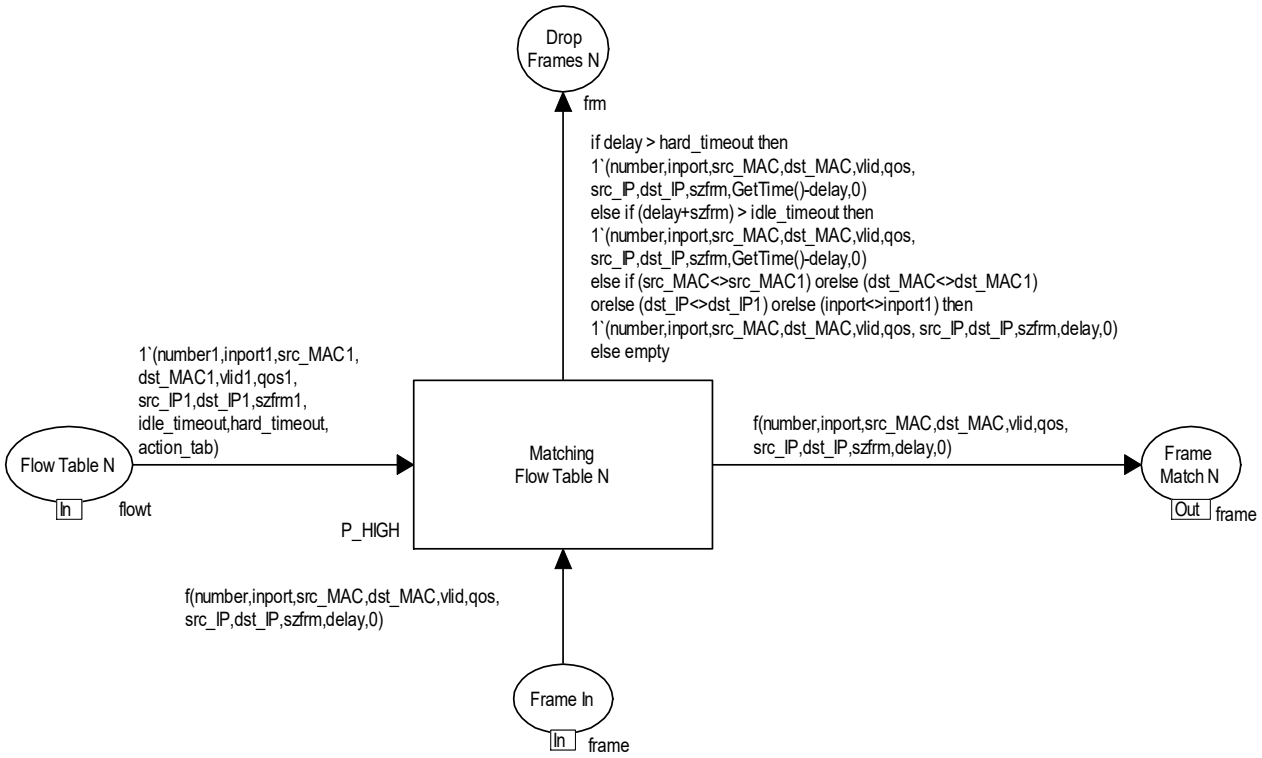

Рисунок 6 – Подсеть сравнения таблицы потоков N
Figure 6 – Subnet of matching of flow table N

Подсеть классификатора Classifier представлена на рисунке 7. Через позиции Frame Match 1, Frame Match N поступает кадр. В позиции Simulation action real-time traffic задается эмуляция действий в коммутаторе с трафиком реального времени. В начальном значении позиции задается выражение: if Range.ran() <= possibility then "set_queue1" else "output_port". Таким образом, эмулируется случайный характер в диапазоне от некоторой константной ве-

личины вероятности. Выполняются команды постановки кадра реального времени в очередь 1 через «set_queue1» либо передача в выходной порт-кадр через «output_port».

В случае, если пришел кадр реального времени с приоритетом qos = 7 и выполняется эмуляция передачи кадра в очередь 1 action1 = "set_queue1", то такой кадр помещается в эту очередь. Подробное описание срабатывания условия представлено ниже:

if (qos = 7) andalso action1 = "set_queue1" then f(number,inport,src_MAC,dst_MAC, vlid, qos, src_IP,dst_IP,szfrm,GetTime(),0) else avail.

Если пришел кадр реального времени и происходит передача кадра в выходной порт согласно таблице поток, то выполняется следующее условие:

if (qos = 7) andalso action1 = "output_port" then 1`(number,inport,src_MAC,dst_MAC, vlid, qos, src_IP,dst_IP,szfrm,GetTime(),0) else empty.

Если поступивший кадр отличается от трафика реального времени, он помещается в очередь 2, поскольку для такого вида трафика не важны задержка кадра, быстродействие передачи в сети. Подробное описание срабатывания условия представлено ниже:

if (qos <> 7) then f(number,inport,src_MAC,dst_MAC,vlid,qos,src_IP, dst_IP, szfrm, GetTime(),0) else avail.

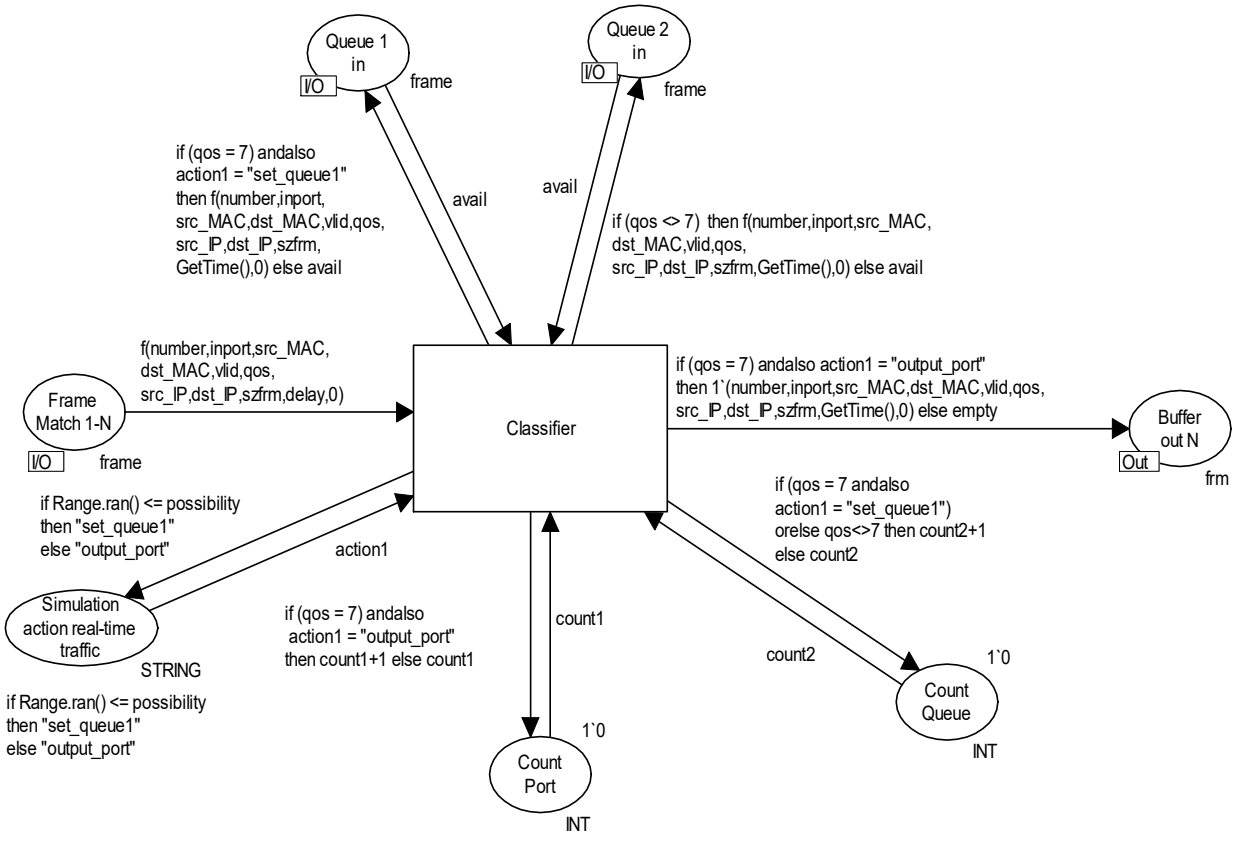

Рисунок 7 – Подсеть классификатора
Figure 7 – Subnet of classifier

Согласно протоколу OpenFlow должна вестись статистика в таблицах потоков через их специальный формат и поля Counters. Счетчики ведут различную статистику поступивших, переданных кадров, ошибочных ситуаций и т.д. Для этих целей были введены позиции подсчета кадров, переданных в очередь любого типа трафика через позицию Count Queue. Условие срабатывания для позиции выглядит следующим образом: if (qos = 7 andalso action1 = "set_queue1") orelse qos<>7 then count2+1 else count2. Статистика переданных кадров в выходной порт ведется в позиции Count Port и описана как: if (qos = 7) andalso action1 = "output_port" then count1+1 else count1.

Подсети очередей 1 и 2, диспетчера используются стандартного вида и описаны в статьях [1-2]. Задержка вычисляется на выходном порте через условие 1@+(96+szfrm+64) на позиции OutPort N Free. Таким образом, в расчете участвуют время межкадрового интервала, метаданные кадра и непосредственно размер кадра szfrm. Подсеть выходного порта коммутатора OpenFlow представлена на рисунке 8.

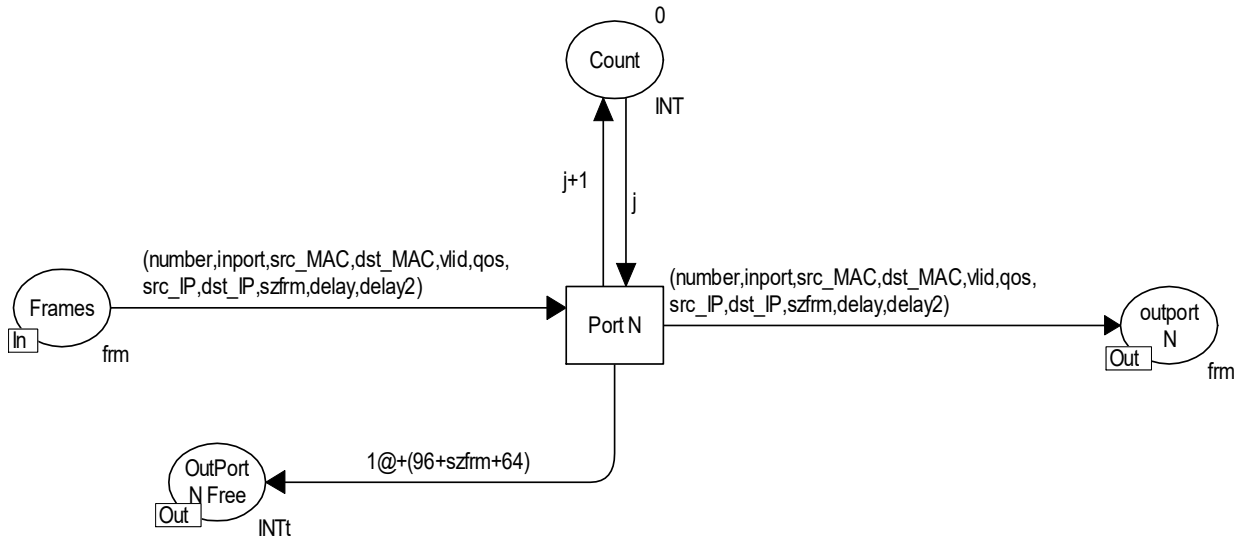

Рисунок 8 – Подсеть выходного порта
Figure 8 – Subnet of output port

С помощью специальных программных средств среды CPN Tools мониторов происходит статистика и запись в файл исследуемого трафика, задержки и удаленных кадров. Монитор в подсети выходного порта на переходе Port N представлен ниже.

```
fun pred (bindelem) = let
  fun predBindElem (OutPort_N'Port_N(1, {inport,delay,delay2,j,number, src_MAC,
dst_MAC,src_IP,dst_IP,qos,szfrm,vlid})) = true
      | predBindElem _ = false
in
  predBindElem bindelem
end
fun obs (bindelem) = let
  fun obsBindElem (OutPort_N'Port_N (1, {inport,delay,delay2,j,number,qos, src_MAC,
dst_MAC, src_IP,dst_IP,szfrm,vlid})) = Int.toString(number)^", "^Int.toString(src_MAC)^",
"^Int.toString(dst_MAC)^", "^Int.toString(vlid)^","^Int.toString(qos)^", "^Int.toString(src_IP)^",
"^Int.toString(dst_IP)^", "^Int.toString(szfrm)^", "^Int.toString(delay)^",
"^Int.toString(delay2)^"++\n"
      | obsBindElem _ = ""
in
  obsBindElem bindelem
end
```

Данный монитор записывает в файл прошедший через коммутатор трафик в файл с временной задержкой. Следующий монитор в коммутаторе хранит учет потерянных кадров в таблицах потоков через переходы Matching Flow Table 1, Matching Flow Table N и описан ниже.

```
fun pred (bindelem) = let
```

```
    fun predBindElem (Matching1'Matching_Flow_Table_1 (1, {action_tab,delay,dst_IP,dst_IP1,dst_MAC,dst_MAC1,hard_timeout,idle_timeout,inport,inport1, number,number1,qos,qos1,src_IP, src_IP1,src_MAC,src_MAC1,szfrm, szfrm1,vlid,vlid1})) = true
        | predBindElem _ = false
    in
     predBindElem bindelem
    end
    fun obs (bindelem) = let
     fun obsBindElem (Matching1'Matching_Flow_Table_1 (1, {action_tab,delay,dst_IP,dst_IP1, dst_MAC,dst_MAC1,hard_timeout,  idle_timeout,inport,inport1,number,number1,qos,qos1,src_IP, src_IP1,src_MAC,src_MAC1,szfrm, szfrm1,vlid,vlid1})) =
    Int.toString(number)^","^Int.toString(src_MAC)^","^Int.toString(dst_MAC)^","^Int.toString(vlid)^","^Int.toString(qos)^",  "^Int.toString(src_IP)^",  "^Int.toString(dst_IP)^",  "^Int.toString(szfrm)^ ",  "^Int.toString(delay)^",0++\n"
        | obsBindElem _ = ""
    in
     obsBindElem bindelem
    end
```

Было проведено экспериментальное исследование разработанной модели на основе аппарата сетей Петри. Исследовался большой трафик для передачи его в ПКС и коммутатору OpenFlow, он равнялся 3621, 5561 и 7843 кадрам. Таким образом, конечная загрузка коммутатора составляла 0.4, 0.6 и 0.8. Рабочей загрузкой коммутатора в промышленной среде является 0.8.

Производились замеры времени передачи потока (кадра) в таблицах потоков и совпадение с одним из правил таблицы. Учитывались замеры времени с вхождения в таблицу потоков 0 до N и сколько времени в тактовых импульсах (ТИ) требовалось для определения нужного правила из таблицы потоков N.

В разработанной модели передачи разнородного трафика в ПКС использовались только две таблицы потоков, но, как известно, их можно увеличить для получения большего эффекта производительности предложенного метода ранней диагностики трафика реального времени. Результаты моделирования представлены в таблице 1.

Таблица 1 – Результат моделирования

Table 1 – Result of modeling

Характеристика	Входящий трафик		
	0,4	0,6	0,8
Количество кадров	3621	5561	7843
Среднее время, через которое удаляется кадр в случае не найденного правила в ПКС, ТИ	5	7	11

В результате, как можно убедиться из таблицы, не представляется возможным определить на ранней стадии анализа передачи трафика его потери в ПКС, и таким образом, не позволит контроллеру оперативно принять решение для повторной передачи трафика реального времени [10], поскольку для данного типа трафика очень важна задержка, и он критичен к ней.

В соответствии с таблицей 1 построен график зависимости количества кадров и среднего времени, через которое удаляется кадр из ПКС, и представлен на рисунке 9.

Как видно в разработанной модели передачи трафика в ПКС, по оценке результатов моделирования сложно диагностировать раннюю потерю трафика в сети из-за специфической организации таблиц потоков и учета тайм-аутов только на поздних этапах контроля передачи трафика. Определенное количество тактов в сети теряется только на анализ потери трафика согласно результатам исследования (рисунок 9), но повторная передача трафика реального времени будет приводить к задержке потоков (кадров), не считая данные тактовые импульсы.

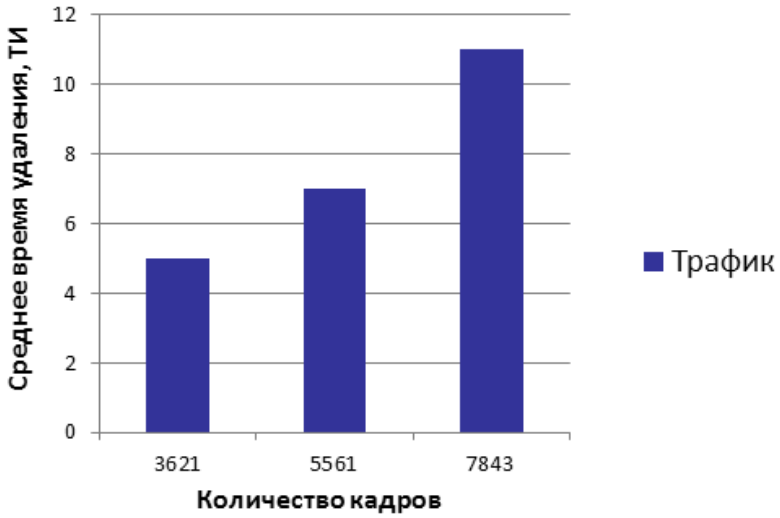

Рисунок 9 – Потери исследуемого трафика в ПКС
Figure 9 – Losses of research traffic in SDN

Заключение

Было проведено моделирование передачи гетерогенного разнородного пользовательского трафика в коммутаторе OpenFlow ПКС на основе цветных временных иерархических сетей Петри и использованием пакета моделирования CPN Tools. Разработаны и описаны подсети чтения заданного трафика, сравнения с правилами кадров в таблицах потоков, классификация трафика и обработка последующих действий с входным трафиком в коммутаторе. Иерархическая модель на сетях Петри позволила не просто исследовать функционирование и поведение ПКС, но и исследовать долю потерянных кадров реального времени в сети, а также задержку.

Библиографический список

1. **Никишин К. И.** Механизм управления трафиком реального времени в коммутаторе Ethernet // Вестник компьютерных и информационных технологий. 2015. № 10. С. 32-37.

2. **Kizilov E., Konnov N., Nikishin K., Pashchenko D., Trokoz D.** Scheduling queues in the Ethernet switch, considering the waiting time of frames. MATEC Web of Conferences. 2016, vol. 44, pp. 01011-p.1–01011-p. 5.

3. **McKeown N., Anderson T., Balakrishnan H. et al.** Openflow: enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review, 2008, vol. 38, no. 2, pp. 69-74.

4. Описание стандарта IEEE 802.1q [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/IEEE_802.1Q

5. **Kobayashi M., Seetharaman S., Parulkar G., Appenzeller G., Little J., Van Reijendam J., McKeown N.** Maturing of OpenFlow and Software-Defined Networking Through Deployments, Computer Networks, 2014, vol. 61, pp. 151-175.

6. **Маньков В. А., Краснова И. А.** Алгоритм динамической классификации потоков в мультисервисной SDN-сети // T-Comm: Телекоммуникации и транспорт. 2017. Том 11. № 12. С. 37-42.

7. **Nikishin K., Konnov N.** Schedule Time-Triggered Ethernet. International Conference on Engineering Management of Communication and Technology, EMCTECH 2020. DOI: 10.1109/EMCTECH49634.2020.9261540.

8. **Никишин К. И, Коннов Н. Н.** Генератор трафика Ethernet на основе цветных сетей Петри. Модели, системы, сети в экономике, технике, природе и обществе. 2016. № 1 (17). С. 299-307.

9. **Никишин К. И.** Моделирование контроллера и верификация процесса передачи данных в программно-конфигурируемых сетях // Вестник Рязанского государственного радиотехнического университета. 2022. № 80. С. 75-83.

10. **Никишин К. И.** Моделирование и верификация топологий программно-конфигурируемых сетей // Вестник Рязанского государственного радиотехнического университета. 2022. № 80. С. 67-74.

UDC 681.31

MODELING OF TRAFFIC TRANSMISSION IN SOFTWARE DEFINED NETWORKS

K. I. Nikishin, Ph.D. (Tech.), lecturer, department of computer science, PSU, Penza, Russia; orcid.org/0000-0001-7966-7833, e-mail: nkipnz@mail.ru

Traffic and criteria management in Software Defined Networks (SDN) is performed by a controller in conjunction with a switch that operates using the OpenFlow protocol. The OpenFlow protocol is the main part of the SDN network. Thus, the OpenFlow protocol successfully allows to cope with the increased requirements for user traffic, priority. ***The aim of the work*** *is to research and model the transmission of heterogeneous user traffic in the OpenFlow switch of SDN based on colored temporary hierarchical Petri nets and using CPN Tools modeling package. The objectives of the research are the development and construction of subnets for reading a given traffic, comparison with frame rules in flow tables, traffic classification and processing of subsequent actions with input traffic in the switch. The hierarchical model on Petri nets allowed us not only to investigate SDN network functioning and behavior, but also to investigate the proportion of lost real-time frames in a network, as well as latency.*

Key words: *Software Defined Networks, controller, switch, Ethernet, OpenFlow, Flow Table, time-outs, Petri Nets, CPN Tools.*

DOI: 10.21667/1995-4565-2022-81-32-41

References

1. **Nikishin K. I.** Mehanizm upravlenija trafikom real'nogo vremeni v kommutatore Ethernet. *Vestnik komp'juternyh i informacionnyh tehnologij*. 2015, no. 10, pp. 32-37. (in Russian).

2. **Kizilov E., Konnov N., Nikishin K., Pashchenko D., Trokoz D.** Scheduling queues in the Ethernet switch, considering the waiting time of frames. *MATEC Web of Conferences*. 2016, vol. 44, pp. 01011-p.1–01011-p. 5.

3. **McKeown N., Anderson T., Balakrishnan H. et al.** Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 2008, vol. 38, no. 2, pp. 69-74.

4. Opisanie standarta IEEE 802.1q [Jelektronnyj resurs] – Rezhim dostupa: https://ru.wikipedia.org/wiki/IEEE_802.1Q. (in Russian).

5. **Kobayashi M., Seetharaman S., Parulkar G., Appenzeller G., Little J., Van Reijendam J., McKeown N.** Maturing of OpenFlow and Software-Defined Networking Through Deployments, Computer Networks, 2014, vol. 61, pp. 151-175.

6. **Man'kov V. A., Krasnova I. A.** Algoritm dinamicheskoj klassifikacii potokov v mul'tiservisnoj SDN-seti. *T-Comm: Telekommunikacii i transport*. 2017, vol. 11, no. 12, pp. 37-42. (in Russian).

7. **Nikishin K., Konnov N.** Schedule Time-Triggered Ethernet. International Conference on Engineering Management of Communication and Technology. *EMCTECH 2020.* DOI: 10.1109/EMCTECH49634.2020.9261540.

8. **Nikishin K. I, Konnov N. N.** Generator trafika Ethernet na osnove cvetnyh setej Petri. *Modeli, sistemy, seti v jekonomike, tehnike, prirode i obshhestve*. 2016, no. 1 (17), pp. 299-307.

9. **Nikishin K. I.** Modelirovanie kontrollera i verifikaciya processa peredachi dannyh v programm-no-konfiguriruemyh setyah. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta*. 2022, no. 80, pp. 75-83. (in Russian).

10. **Nikishin K. I.** Modelirovanie i verifikaciya topologij programmno-konfiguriruemyh setej. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta*. 2022, no. 80, pp. 67-74. (in Russian).