Научная статья
УДК 004.72
https://doi.org/10.25686/2306-2819.2022.2.47

Моделирование метода ранней диагностики потерь трафика реального времени в программно-конфигурируемых сетях на основе аппарата сетей Петри

К. И. Никишин
Пензенский государственный университет,
Российская Федерация, 440026, Пенза, ул. Красная, 40
nkipnz@mail.ru

Аннотация. *В статье рассмотрена и исследована модель метода ранней диагностики потерь трафика реального времени в программно-конфигурируемых сетях (ПКС) при передаче разнородного трафика в коммутаторе OpenFlow на основе цветных временных иерархических сетей Петри и с использованием пакета моделирования CPN Tools. Разработаны и описаны подсети Петри, моделирующие основной процесс передачи трафика в ПКС. Проведено экспериментальное моделирование с различной загрузкой коммутатора и сравнение результатов разработанного метода с классическим методом передачи разнородного трафика в ПКС.*

Ключевые слова: *программно-конфигурируемые сети; трафик реального времени; контроллер; коммутатор; Ethernet; OpenFlow; таблицы потоков; таймауты; сети Петри; CPN Tools*

Введение. Современные информационные технологии, программные продукты развиваются стремительным образом. Данное определение подходит и к компьютерным сетям [1]. Классической компьютерной сетью является Ethernet с поддержкой «качества обслуживания» (Quality of Service QoS). QoS позволяет передавать разнородный трафик в зависимости от его приоритета, тем самым в коммутаторе Ethernet происходит классификация трафика [2–3].

Дальнейшим развитием компьютерных сетей стало появление распределённых сетей, таких как Time-Triggered Ethernet [4], программно-конфигурируемые сети (ПКС) [5–7], облачные вычисления. Причиной перехода к развитию распределённых сетей стало ограничение классической компьютерной сети Ethernet по дальнейшим разработкам и вариациям передачи трафика согласно стандарту IEEE 802.1 [8]. Тем самым накладываются новые ограничения по быстродействию трафика реального времени, разбросу средней задержки на выходе коммутатора (джиттер) и отказоустойчивости сети [9–11].

В ПКС основным протоколом является OpenFlow, протокол позволяет обрабатывать разнородный трафик и управлять им [12–14]. Схематично обработку трафика в ПКС можно описать следующим образом: поступающий трафик перенаправляется на вход одного из портов коммутатора OpenFlow [15]. Трафик поступает с уровня инфраструктуры на уровень управления.

К основным недостаткам протокола OpenFlow можно отнести значительное увеличение времени поиска правила для кадра в таблицах потоков. Это приводит

Для цитирования: Никишин К. И. Моделирование метода ранней диагностики потерь трафика реального времени в программно-конфигурируемых сетях на основе аппарата сетей Петри // Вестник Поволжского государственного технологического университета. Сер.: Радиотехнические и инфокоммуникационные системы. 2022. № 2 (54). С. 47-60. DOI: https://doi.org/10.25686/2306-2819.2022.2.47

к информированию контроллера о необходимости удаления кадра из ПКС на более позднем этапе диагностики. Поздняя диагностика потерь трафика происходит из-за поиска правила во всех таблицах потоков, и в случае если не будет найдено необходимое правило в таблице, только тогда удаляется кадр из ПКС.

Однако контроллеру может потребоваться больше времени для удаления кадра из ПКС за счёт различных архитектур и топологий ПКС [16], где возможна группа иерархий контроллеров. В таком случае время на принятие решения для повторной передачи трафика может быть увеличено, поскольку в передаче участвуют уже несколько контроллеров.

К следующему недостатку протокола OpenFlow можно отнести значительные аппаратные затраты на ресурсы по хранению таблиц потоков. Тем самым возможно упростить структуру хранения таблиц за счёт вынесения функций контроля таймаутов из таблиц потоков на входы коммутатора OpenFlow.

Таким образом, автор статьи предложил метод ранней диагностики потерь трафика реального времени с контролем таймаутов в ПКС. На входящий порт коммутатора устанавливается аппаратный защитник по контролю таймаутов. Защитник контролирует только трафик реального времени, стохастический трафик передаётся и управляется стандартным образом по протоколу OpenFlow.

Тем самым уже на входе коммутатора производится анализ типа трафика: реального времени или стохастический. Если трафик реального времени поступает на вход защитника контроля таймаутов и выполняются все успешные проверки таймаутов, то такой кадр продвигается далее в коммутаторе.

Если не выполнится контроль какого-либо таймаута, то происходит удаление кадра из ПКС и информирование контроллера ПКС о повторной передаче кадра. Метод позволяет определить более раннюю диагностику потерь трафика реального времени, чем классическим методом по таблице потоков.

Цель исследования – повышение эффективности диагностики потерь трафика реального времени в ПКС.

Задачами исследования являются построение классификации типа трафика на входе коммутатора, построение подсети Петри процесса передачи трафика реального времени к подсетям контроля таймаутов в ПКС, построение подсети Петри контроля hard timeout (принудительное удаление записи из таблицы и кадра в заданное время), построение подсети Петри контроля idle timeout (удаление записи из таблицы и кадра в случае не достижения передачи приёмной стороны), модификация подсетей таблицы потоков, поиск правила в таблицах потоков, экспериментальное моделирование с различной загрузкой коммутатора и сравнение результатов разработанного метода с классической передачей разнородного трафика в ПКС.

Моделирование метода. Разработана и описана модель метода ранней диагностики потерь трафика реального времени в ПКС на основе цветных временных иерархических сетей Петри [17] с помощью свободно распространяемого пакета CPN Tools, который наилучшим образом подходит для исследования компьютерных сетей, их критериев, метрик и задержки в сети. Существуют и другие специализированные сетевые пакеты для имитационного моделирования сетей [18], но CPN Tools обладает максимальной производительностью по исследованию компьютерных сетей.

Вначале происходит чтение разнородного трафика, который подразумевает собой сочетание трафика реального времени и стохастического (эластичного) трафика. Разграничение по типу трафика производится согласно полю в формате кадра Ethernet IEEE 802.1, и отвечает за это поле качества обслуживания. Более подробное описание работы генератора разнородного трафика, а также чтение сгенерированного трафика представлено в статье [19].

Подсеть буфера коммутатора OpenFlow представлена на рис. 1. С помощью перехода Control Real-time frame происходит разграничение типа трафика для дальнейшей обработки с помощью контроля таймаутов в защитнике ПКС. Через позицию Real-time передаётся трафик реального времени следующим образом: if qos = 7 then f(number, inport, src_MAC, dst_MAC, vlid, qos, src_IP, dst_IP, szfrm, GetTime(), delay2) else avail. Таким образом, если поле qos будет отличаться от максимального приоритета 7, то такой кадр передаётся в позицию Input frame (эластичный кадр).

В случае с трафиком реального времени он направляется в подсеть Transmitting frame, которая отвечает за передачу кадра от входного порта к блокам контроля задержек. Процесс передачи осуществляется не мгновенно в модели, так же как и в промышленной среде, а с некоторой дополнительной небольшой задержкой.

Подсеть процесса передачи кадра реального времени Transmitting frame представлена на рис. 2. Входной кадр передаётся через позицию Input frame и переход Transmitting from SDN controller в позицию Frame, где фиксируется время прибытия кадра в формате самого кадра через функцию GetTime().

Переход Get transmitted time выполняет вычисление небольшой величины времени передачи кадра, и значение записывается в позицию Transmitted time, описывается следующим образом: if Range.ran() <= percent2 then 1`GetTime() else 1`(GetTime()-500).

Данное время необходимо для эмуляции потерь трафика в сети. В одном случае потерь не будет в сети, поскольку время прибытия кадра будет равно текущему времени локальной машины, в другом случае будет эмулироваться потеря трафика за счёт уменьшения времени прибытия кадра на константу 500.

Кадр из позиции Frame to hard time и время передачи Transmitted time направляются в подсеть контроля Control hard time. Подсеть Control hard time будет описана ниже. В случае успешных проверок таймаутов по протоколу OpenFlow кадр передаётся в позицию Frame to OpenFlow из подсети Hard time. После этого кадр возвращается через выходную позицию подсети Real-time frame out в позицию Input frame подсети буфера коммутатора OpenFlow, описывается следующим выражением: if qos = 7 then f(number,inport,src_MAC, dst_MAC, vlid, qos, src_IP,dst_IP, szfrm,GetTime(),delay2) else avail.

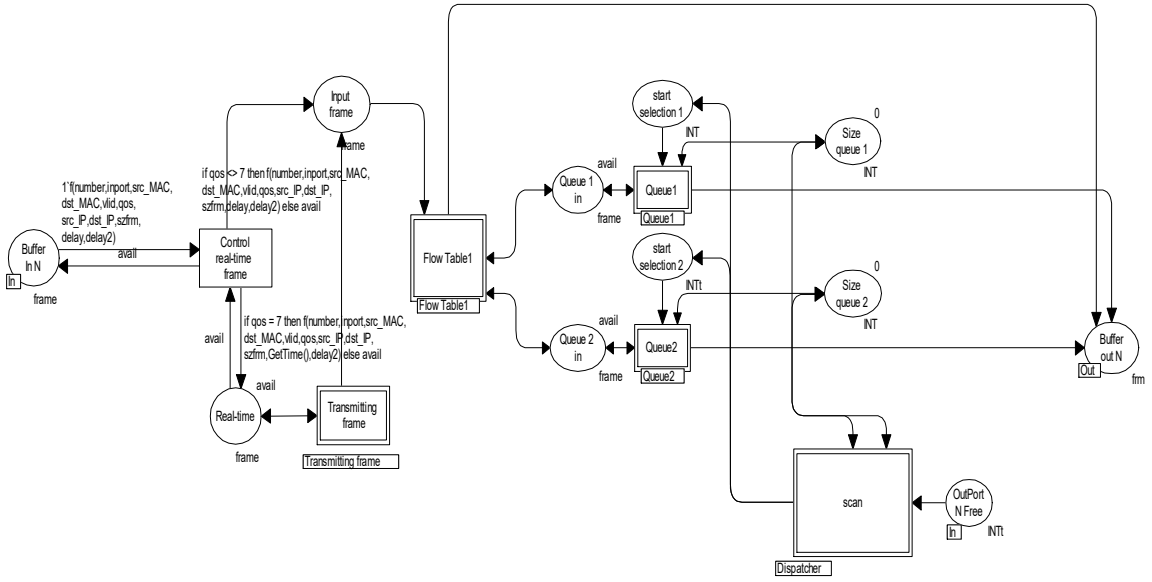

Рис. 1. Подсеть буфера коммутатора OpenFlow согласно предложенному методу
Fig. 1. OpenFlow switch buffer subnet according to the proposed method

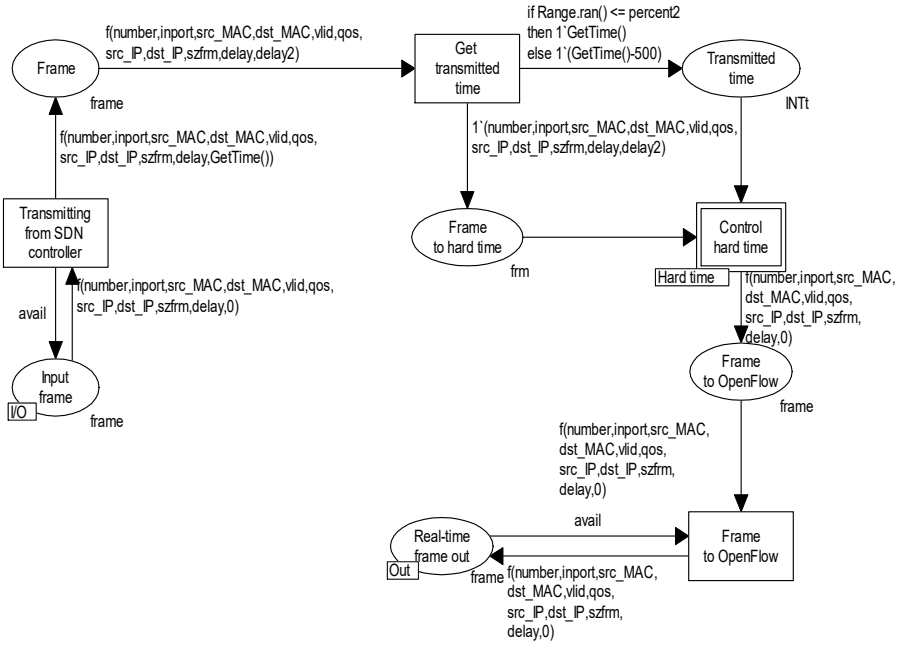

Рис. 2. Подсеть процесса передачи кадра реального времени Transmitting frame
Fig. 2. Transmitting frame subnetframe

Подсеть Hard time представлена на рис. 3, она контролирует жёсткий таймаут на входе коммутатора OpenFlow. Входящий кадр из позиции Input frame поступает через переход Get receive time в позицию Receive time и после в позицию Frame.

В подсети контроля hard timeout вычисляются время начала и окончания временного окна таймаута. В качестве начала временного окна отвечает позиция Begining time, равная времени передачи кадра. Для вычисления окончания временного окна необходима настройка таймаута для подстройки и синхронизации часов между коммуникационными узлами.

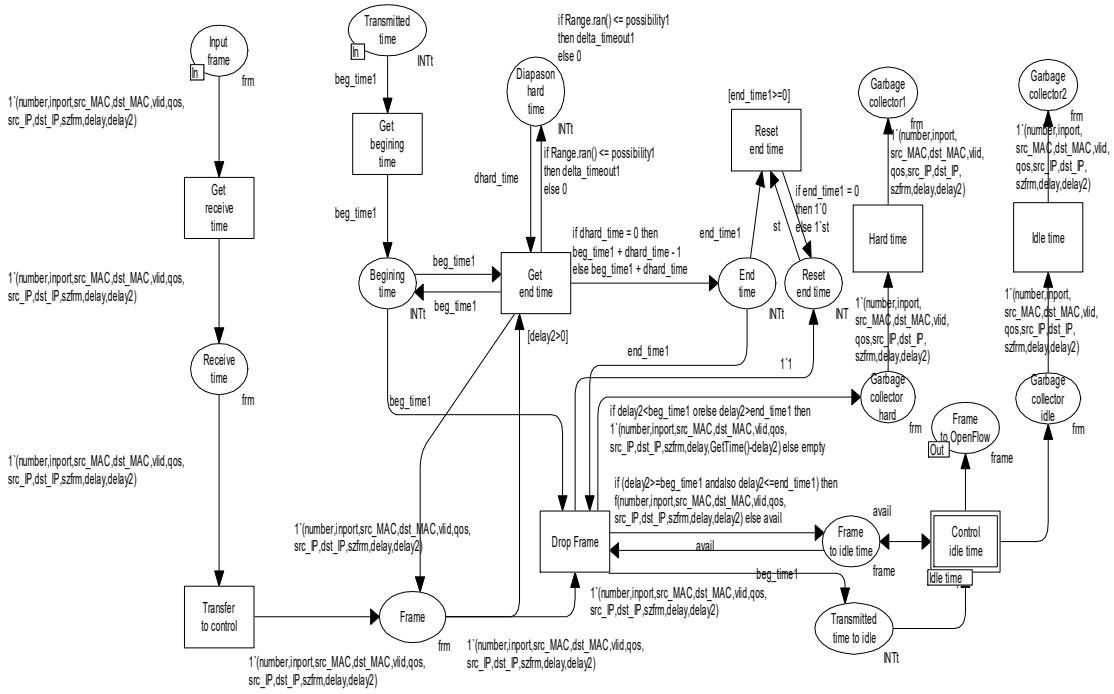

Рис. 3. Подсеть контроля hard timeout
Fig. 3. Hard timeout control subnet

Настройка рассчитывается в позиции Diapason hard time в виде начального значения с помощью выражения: if Range.ran() <= possibility1 then delta_timeout1 else 0. Настройка может принимать следующие значения: 0 (не требуется настройка) или константа delta_timeout1, определённая в модели CPN Tools.

Переход Get end time выполняет вычисление времени окончания окна. На вход перехода поступают переменные beg_time1 (время начала окна), dhard_time (настройка таймаута) и входящий кадр. Переход Get end time срабатывает только в том случае, если delay2>0 во входящем кадре. В позицию End time записывается время окончания окна контроля, и выражение выглядит следующим образом: if dhard_time = 0 then beg_time1 + dhard_time – 1 else beg_time1 + dhard_time.

Если не требуется настройка величины таймаута и необходима эмуляция удаления кадра из ПКС, то уменьшается время окончания окна, в противном случае добавляется настройка таймаута для увеличения времени окончания окна.

Через позицию Drop Frame удаляются кадры или передаются далее для дальнейшего контроля кадра. В позицию Frame to idle time кадр поступает следующим образом: if (delay2>=beg_time1 andalso delay2<=end_time1) then f(number, inport, src_MAC, dst_MAC, vlid, qos, src_IP, dst_IP, szfrm, delay, delay2) else avail. Таким образом, время прибытия кадра реального времени должно быть больше или равно времени начала окна контроля и меньше или равно времени окончания окна контроля. В случае несоответствия данного выражения кадр передаётся на удаление с помощью выражения: if delay2<beg_time1 orelse delay2>end_time1 then 1` (number,inport,src_MAC, dst_MAC, vlid, qos, src_IP, dst_IP, szfrm, delay, GetTime()-delay2) else empty.

В позицию Transmitted time to idle передаётся исходное время передачи кадра для подсети контроля idle timeout. Подсчёт удалённых кадров из ПКС ведётся с помощью встроенного инструмента монитора среды CPN Tools. Для этих целей на переход Hard time был введён монитор, он описан ниже.

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

Удаление кадров происходит с помощью функции очистки в позициях Garbage collector1 контроля таймаута hard timeout и Garbage collector2 контроля таймаута idle timeout. Подсеть Idle time представлена на рис. 4 и контролирует таймаут idle timeout.

Отличительной особенностью данной подсети от подсети на рис. 3 является то, что учитывается время передачи кадра с учётом его длины. Таким образом, к величине времени окончания окна контроля idle timeout добавляется размер кадра, и условие выглядит следующим образом: if d_idle_time = 0 then beg_time1 – 1 else beg_time1 + d_idle_time + szfrm.

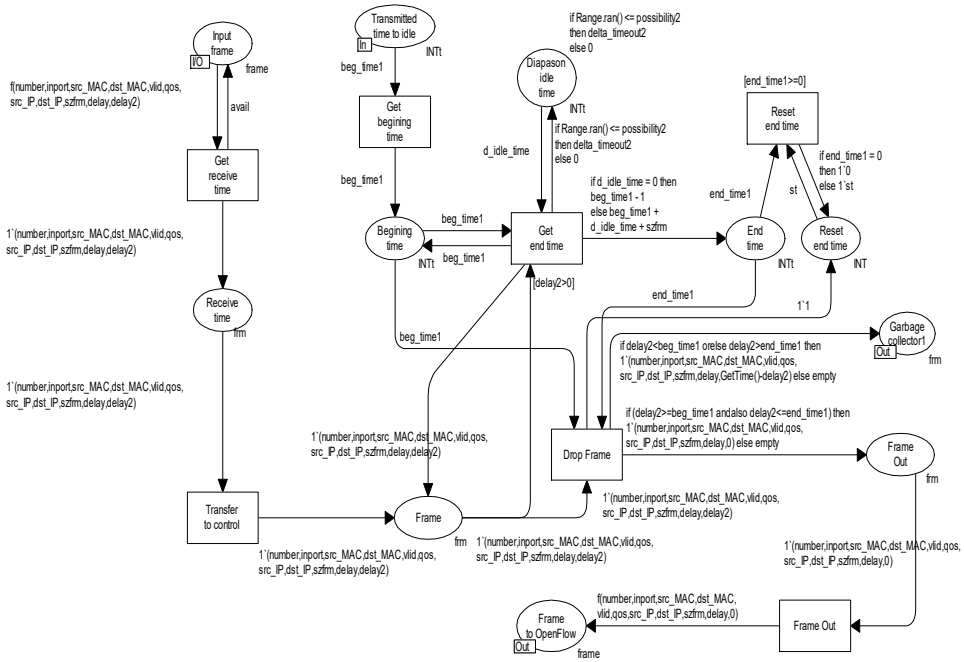

Рис. 4. Подсеть контроля idle timeout
Fig. 4. Idle timeout control subnet

Реализация монитора на переходе Drop Frame представлена ниже.

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

Разработанные подсети контроля таймаутов hard timeout и idle timeout распознают на ранней стадии потерю кадров трафика реального времени, и это обстоятельство позволяет информировать контроллер ПКС о повторной передаче кадра. Анализ потерянных кадров производился на основе полученных файлов потерь трафика реального времени. Файлы были записаны с помощью описанных выше мониторов.

В связи с введением отдельных подсетей контроля таймаутов на входе коммутатора значительно упрощается передача трафика по таблицам потоков, поскольку отсутствует контроль хранения и обработки таймаутов по протоколу OpenFlow. На рис. 5 представлена таблица потоков 1.

Кроме этого, упрощается выражение в позиции Flow Table OpenFlow, которое отвечает за хранение набора правил по OpenFlow, и оно выглядит следующим образом:

1`(number,inport,src_MAC,dst_MAC,vlid,qos,src_IP,dst_IP,szfrm,GetTime() + zfrm+delta_timeout1,GetTime()+delta_timeout1,"to_port/to_queue/drop"). Здесь уже не учитываются потери трафика реального времени, как в классическом методе передачи ПКС, т. к. потери были определены ещё на входе коммутатора.

Модернизирована подсеть сравнения и поиска правила в таблице потоков 1, подсеть представлена на рис. 6. Отсутствует позиция Drop Frames 1, которая учитывала заданные таймауты и удаляла кадр. Кроме этого, введена позиция Frame Match 1-N, в которую передаётся кадр для осуществления поиска правила кадра в таблице потоков 1, и выражение выглядит следующим образом: if qos = 7 andalso mode = 1 then f(number, inport, src_MAC, dst_MAC, vlid, qos, src_IP, dst_IP, szfrm, delay + delta_timeout1, 0) else f(number, inport, src_MAC, dst_MAC, vlid, qos, src_IP, dst_IP, szfrm, delay,0).

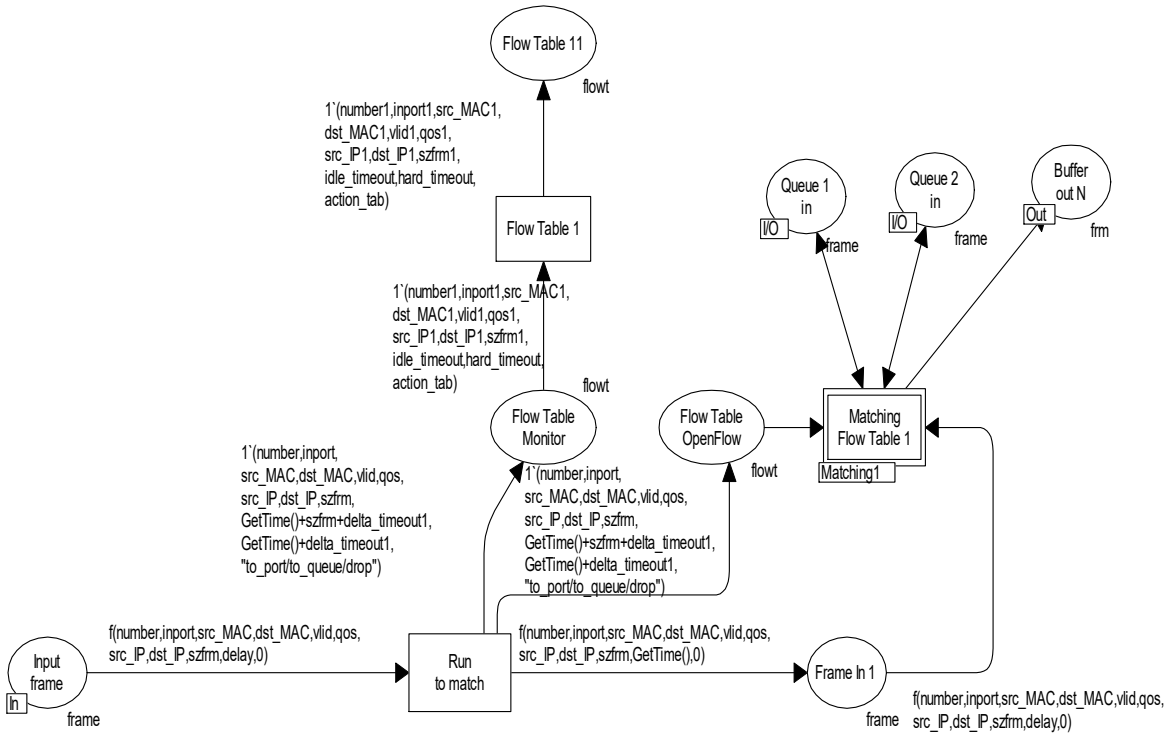

Рис. 5. Подсеть таблица потоков 1
Fig. 5. Subnet flow table 1

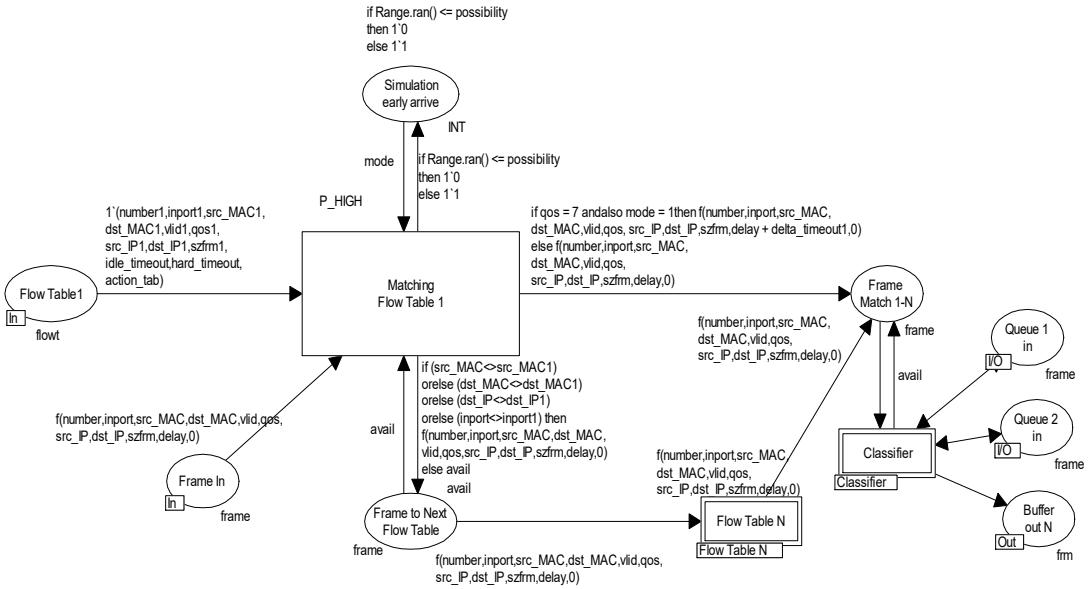

Рис. 6. Подсеть сравнения и поиска правила в таблице потоков 1
Fig. 6. Subnet for comparison and searching rules in flow table 1

Если переменная режима работы установлена mode = 1, то осуществляется режим раннего прибытия кадра в ПКС. Такой кадр направляется в очередь коммутатора, и кадр будет ожидать времени своей передачи. В противном случае кадр прибыл в строго отведённое для него время и будет направлен сразу же в выходной порт коммутатора. Учёт времени прибытия кадра происходит в предпоследнем параметре формата самого кадра и определяется величинами delay + delta_timeout1 или delay. Позиция Simulation early arrive осуществляет эмуляцию раннего прибытия кадра реального времени.

В случае если не найдено правило для кадра в заданной таблице, то он переходит в следующую подсеть таблицы потоков N, где происходит дальнейший поиск и сравнение с правилами в следующих таблицах. Подсеть представлена на рис. 7. В ней также происходит упрощение и отказ от предложенных таймаутов.

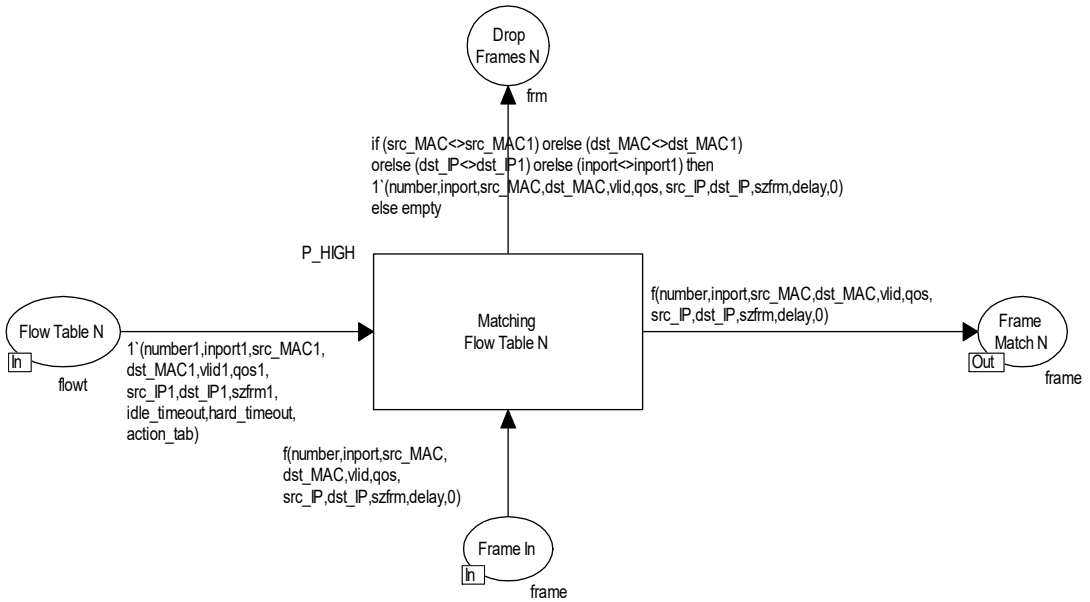

Рис. 7. Подсеть сравнения и поиска правила в таблице потоков N
Fig. 7. Subnet for comparison and searching rules in the flow table N

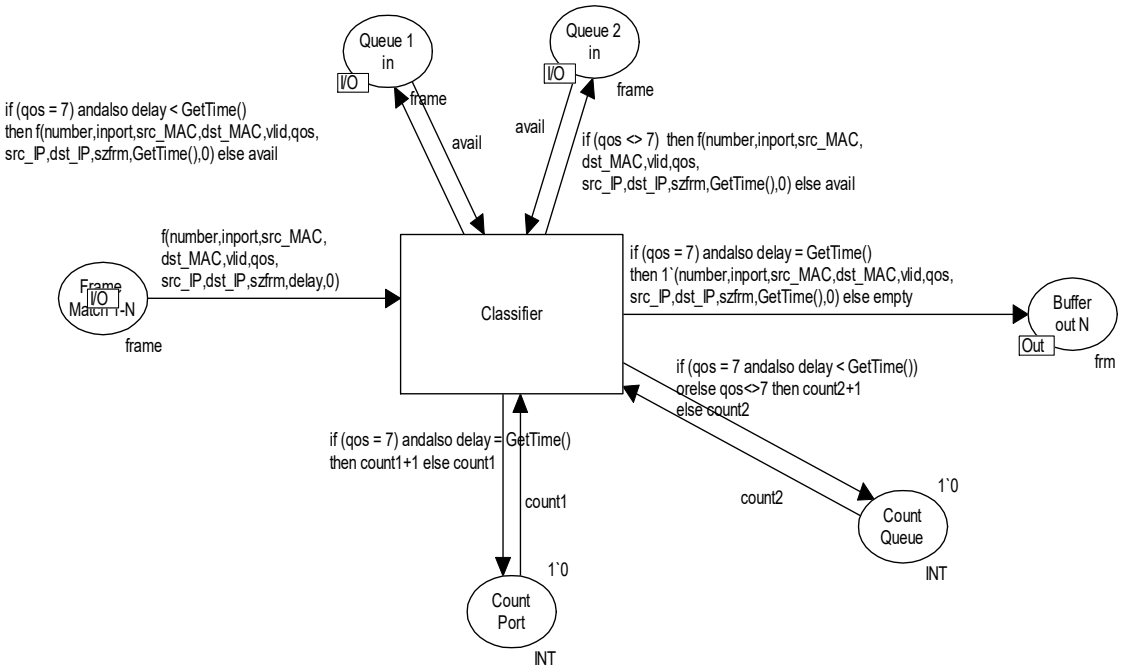

Рис. 8. Подсеть классификации трафика в ПКС
Fig. 8. Traffic classification subnet in software-defined networks

В модели модифицируется подсеть классификации трафика в отличие от классической модели, поскольку вычисление раннего прибытия кадра реального времени происходит уже в подсети сравнения и поиска правила в таблице потоков. Поэтому удаляется позиция Simulation action real-time traffic, которая отвечала за выполнение действия над кадром: поставить кадр в очередь коммутатора, передать кадр сразу в выходной порт. Подсеть классификации трафика представлена на рис. 8.

Кроме этого в подсети классификации трафика учитываются временные характеристики трафика, а не инструкции из таблицы потоков. В очередь 1 направляется трафик реального времени, если время прибытия кадра меньше системного времени локальной машины: if (qos = 7) andalso delay < GetTime() then f(number, inport, src_MAC, dst_MAC, vlid, qos, src_IP, dst_IP, szfrm, GetTime(), 0) else avail.

Если же прибыл эластичный трафик, то он сразу же переходит в очередь 2, поскольку он не критичен к задержке кадра и джиттеру в сети: if (qos <> 7) then f(number, inport, src_MAC, dst_MAC, vlid, qos, src_IP, dst_IP, szfrm, GetTime(), 0) else avail.

Если же время прибытия кадра реального времени будет равно системному времени, то кадр передаётся сразу же в выходной порт, условие выглядит следующим образом: if (qos = 7) andalso delay = GetTime() then 1`(number, inport, src_MAC,dst_MAC,vlid,qos,src_IP,dst_IP,szfrm,GetTime(),0) else empty.

Результаты моделирования. Было проведено экспериментальное исследование и сравнение разработанных моделей на основе аппарата сетей Петри. Исследовался большой трафик для передачи его по ПКС и коммутатору OpenFlow, он равнялся 3 621, 5 561 и 7 843 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 при поиске правила для кадра в одной из таблиц. По найденному правилу искались сгенерированные с этим потоком значения таймаутов hard timeout и idle timeout.

Эксперименты с замерами времени производились по максимальному пути передачи в случае, когда кадр передавался из таблицы потоков от 0 до N. Расчёты приведены в тактовых импульсах (ТИ) системы CPN Tools.

В классической модели передачи разнородного трафика в ПКС использовались только две таблицы потоков для упрощения исследования передачи трафика. Но, как известно, количество таблиц потоков можно увеличить для получения большей эффективности предложенного метода ранней диагностики трафика реального времени. Результаты сравнения методов представлены в таблице.

Результаты моделирования
Simulation results

Характеристика	Входящий трафик		
	0,4	0,6	0,8
Количество кадров	3 621	5 561	7 843
Среднее время, через которое удаляется кадр классическим методом передачи в ПКС, ТИ	5	7	11
Среднее время, через которое удаляется кадр предложенным методом передачи в ПКС, ТИ	3	5	7

Максимальный эффект метода ранней диагностики потерь достигается при рабочей загрузке коммутатора, равной 0,8, и передачи большого количества кадров по сети 7 843. Классическим методом передачи трафика в ПКС тратится в среднем 11 тактовых импульсов для того, чтобы удалить кадр из сети и проинформировать контроллер о повторной передаче кадра трафика реального времени. Предложенный метод ранней диагностики потерь трафика реального времени позволяет определить потерю трафика в среднем через 7 тактовых импульсов.

Таким образом, в ходе экспериментов можно убедиться по таблице и тактовым импульсам, что происходит раннее определение потери трафика на 36 % при нормальной загрузке коммутатора 0,8 через 7 тактовых импульсов, а не тратится 11 тактовых импульсов, как в модели классического метода передачи. Тем самым, за счёт более раннего информирования контроллера будет происходить и снижение задержки кадров, ведь для трафика реального времени задержка критична.

По таблице построен график зависимости среднего времени, через которое удаляется кадр из ПКС, от количества переданных кадров. График представлен на рис. 9. Результаты моделирования трафика были получены с помощью мониторов.

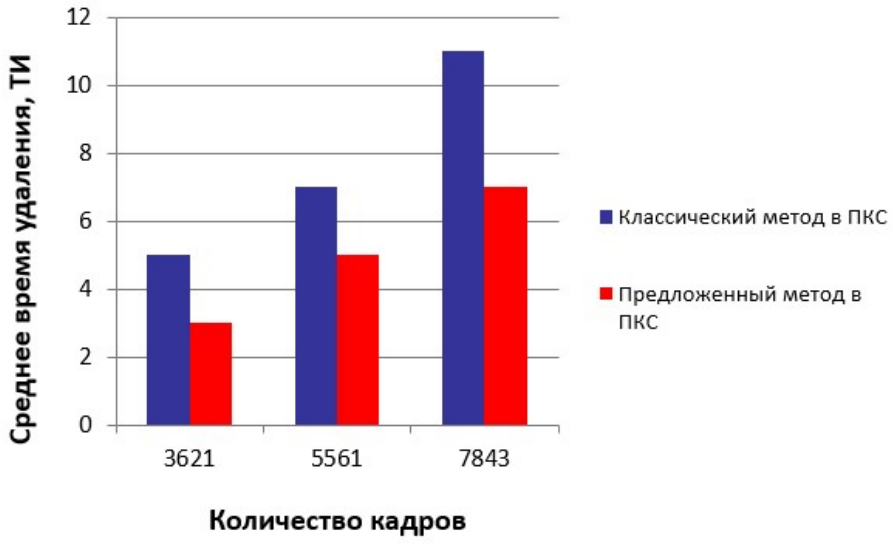

Рис. 9. Среднее время удаления кадра из ПКС
Fig. 9. Average Frame Deletion Time for Software Defined Networks

Заключение. Была разработана и исследована модель метода ранней диагностики потерь трафика реального времени при передаче разнородного трафика в коммутаторе OpenFlow на основе цветных временных иерархических сетей Петри и с использованием пакета моделирования CPN Tools.

Были разработаны и описаны подсети Петри процесса передачи трафика реального времени к блокам контроля таймаутов в ПКС hard timeout и idle timeout, произведена модификация подсетей таблиц потоков, модификация подсети сравнения и поиска правила в таблице потоков.

Было проведено экспериментальное моделирование с различной загрузкой коммутатора, приведено сравнение результатов разработанного метода с классическим методом передачи разнородного трафика в ПКС.

Согласно полученным данным, метод ранней диагностики потерь трафика реального времени в ПКС доказал свою эффективность и обеспечивает раннюю диагностику потерь трафика реального времени. На 36 % тратится меньше времени в тактовых импульсах по сравнению с классическим методом передачи и нормальной загрузкой коммутатора 0,8, что в свою очередь приводит и к снижению задержки при повторной передаче трафика реального времени за счёт раннего информирования контроллера ПКС.

Список источников

1. *Олифер В.Г., Олифер Н.А.* Компьютерные сети. Принципы, технологии, протоколы. 4-е изд. СПб.: Питер, 2010. 943 с.

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

3. Scheduling queues in the Ethernet switch, considering the waiting time of frames / E. Kizilov, N. Konnov, K. Nikishin et al // MATEC Web of Conferences. 2016. Vol. 44. P. 01011-p.1–01011-p. 5.

4. *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.

5. Advanced study of SDN/OpenFlow controllers / A. Shalimov et al. // Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia. ACM, 2013.

6. *Перепелкин Д. А.* Концептуальный подход динамического формирования трафика программно-конфигурируемых телекоммуникационных сетей с балансировкой нагрузки // Информационные технологии. 2015. Т. 21. № 8. С. 602-610.

7. *Перепелкин Д. А., Бышов В. С.* Балансировка потоков данных в программно-конфигурируемых сетях с обеспечением качества обслуживания сетевых сервисов // Радиотехника. 2016. № 11. С. 111-119.

8. Описание стандарта IEEE 802.1q [Электронный ресурс]. URL: https://ru.wikipedia.org/-wiki/IEEE_802.1Q.

9. *Ушакова М. В., Ушаков Ю. А.* Исследование сети виртуальной инфраструктуры центра обработки данных с гибридной программно-конфигурируемой коммутацией // Вестник Рязанского государственного радиотехнического университета. 2021. № 75. С. 34-43. DOI: 10.21667/1995-4565-2021-75-34-43.

10. *Никульчев Е. В., Паяин С. В., Плужник Е. В.* Динамическое управление трафиком программно-конфигурируемых сетей в облачной инфраструктуре // Вестник Рязанского государственного радиотехнического университета. 2013. № 3 (45). С. 54-57.

11. *Леохин Ю. Л., Фатхулин Т. Д.* Оценка возможности предоставления гарантированной скорости передачи данных в программно-конфигурируемой оптической сети // Вестник Рязанского государственного радиотехнического университета. 2020. № 71. С. 45-59. DOI: 10.21667/1995-4565-2020-71-45-59.

12. Программная инфраструктура и визуальная среда распределенной обработки потоков данных в программно-конфигурируемых сетях / В. П. Корячко, Д. А. Перепелкин, М. А. Иванчикова и др. // Вестник Рязанского государственного радиотехнического университета. 2018. № 65. С. 44-54. DOI: 10.21667/1995-4565-2018-65-3-44-54.

13. Openflow: enabling innovation in campus networks / N. McKeown, T. Anderson, H. Balakrishnan et al. // ACM SIGCOMM Computer Communication Review. 2008. Vol. 38, no. 2. Pp. 69-74.

14. Maturing of OpenFlow and Software-Defined Networking Through Deployments / M. Kobayashi, S. Seetharaman, G. Parulkar et al // Computer Networks. 2014. Vol. 61. Pp. 151-175.

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

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

17. *Wael Hosny Fouad Aly.* A novel controller placement using Petri-nets for SDNs, Wseas Transactions on Mathematics. 2020. Vol. 19. Pp. 598-605.

18. *Никишин К.И.* Моделирование беспроводной сенсорной сети с использованием OMNET++ // Вестник Рязанского государственного радиотехнического университета. 2021. № 78. С. 46-54.

19. *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.

Статья поступила в редакцию 25.05.2022; одобрена после рецензирования 10.06.2022; принята к публикации 15.06.2022

Информация об авторе

НИКИШИН Кирилл Игоревич – кандидат технических наук, старший преподаватель кафедры вычислительной техники, Пензенский государственный университет. Область научных интересов – компьютерные сети; передача трафика; телекоммуникационные системы. Автор 47 научных публикаций.

Автор заявляет об отсутствии конфликта интересов.
Автор прочитал и одобрил окончательный вариант рукописи.

Scientific article
UDC 004.72
https://doi.org/10.25686/2306-2819.2022.2.47

Modeling of the Method of Early Diagnosis of Real-Time Traffic Losses in Software Defined Networks Based on the Apparatus of Petri Nets

K. I. Nikishin
Penza State University,
40, Krasnaya str., Penza, 440026, Russian Federation
nkipnz@mail.ru

Keywords*: software defined network; real-time traffic; controller; switch; Ethernet; OpenFlow; flow tables; timeouts; Petri nets; CPN Tools*

ABSTRACT

Introduction*. The classic computer network is Ethernet with support for Quality of Service (QoS). The further development of computer networks was the emergence of distributed networks, such as Time-Triggered Ethernet, software defined networks (SDN), cloud computing. The reason for the transition to the development of distributed networks was the limitation of the classical Ethernet computer network for further developments and variations in traffic transmission according to the IEEE 802.1 standard. In SDN the main protocol is OpenFlow, the protocol allows you to process and manage heterogeneous traffic. The main disadvantages of the OpenFlow protocol include a significant increase in the frame search time in the flow tables and a message to the controller about its removal at a late stage of diagnostics until a search occurs in all flow tables, in case of its inconsistency in fields.* ***The aim of the research*** *is improving the efficiency of loss diagnostics of real-time traffic in SDN.* ***Tasks****: the construction of a classification of the type of traffic at the switch input, the construction of a Petri subnet of the process of transmitting real-time traffic to the timeout controls in the SDN, the construction of a Petri subnet of the control of hard and idle timeouts, modification of the subnets of the flow table, comparison of the flow with the flow table, experimental modeling with different switch loading and comparison of the results of the developed method with the classical transfer of heterogeneous traffic to the SDN.* ***Methods****. A model of the method of early diagnosis of real-time traffic losses in the SDN based on color time hierarchical Petri nets using a free distributed package CPN Tools has been developed and described. The hard timeout and idle timeout control subnets developed recognize the loss of real-time traffic frames at an early stage, and this circumstance allows you to inform the SDN controller about the retransmission of the frame. In connection with the introduction of timeout control by separate subnets, traffic transmission through flow tables is greatly simplified, since there is no control of storage, processing of timeouts using the OpenFlow protocol.* ***Results.*** *An experimental study and comparison of the developed models based on the apparatus of Petri nets was carried out. Measurements of the time from entering the flow table 0 to N and how much time in clock pulses was required to determine the desired rule from the flow table were taken into account. According to the data obtained, the method of early diagnosis of real-time traffic losses in the SDN has proven its effectiveness and provides a reduction in the delay for transmitting real-time traffic with normal switch load 0.8 by 36% on average, the delay of real-time traffic is reduced due to early diagnosis of timeouts in the SDN.*

REFERENCES

1. Olifer V.G., Olifer N.A. *Kompjuternye seti. Principy, tehnologii, protokoly. 4-e izd.* [Computer networks. Principles, technologies, protocols 4th ed.]. Saint Petersburg, Piter, 2010. 943 p. (In Russ.)

2. Nikishin K. I. Mehanizm upravlenija trafikom real'nogo vremeni v kommutatore Ethernet [The mechanism of management real-time traffic in the switch Ethernet]. *Vestnik komp'juternyh i informacionnyh tehnologij* [Herald of computer and information technologies]. 2015. No 10. Pp. 32–37. (In Russ.)

3. Kizilov E., Konnov N., Nikishin K. et al. Scheduling queues in the Ethernet switch, considering the waiting time of frames. *MATEC Web of Conferences*. 2016. Vol. 44. P. 01011-p.1–01011-p. 5.

4. 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.

5. Shalimov A. et al. Advanced study of SDN/OpenFlow controllers. *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russi*a. ACM, 2013.

6. Perepelkin D. A. Konceptualnyj podhod dinamicheskogo formirovanija trafika programmno-konfiguriruemyh telekommunikacionnyh setej s balansirovkoj nagruzki. Informacionnye tehnologii [Conceptual approach to dynamic traffic shaping of software-defined telecommunications networks with load balancing]. *Informacionnye tehnologii* [Information technologies]. 2015. Vol. 21. No 8. Pp. 602-610. (In Russ.)

7. Perepelkin D. A., Byshov V. S. Balansirovka potokov dannyh v programmno-konfiguriruemyh setjah s obespecheniem kachestva obsluzhivanija setevyh servisov [Load balancing in software defined networks with quality of services]. *Radiotehnika* [Radioengineering]. 2016. No 11. P. 111-119. (In Russ.)

8. IEEE 802.1Q. Access: https://ru.wikipedia.org/wiki/IEEE_802.1Q (date of reference: 10.10.2022)

9. Ushakova M. V., Ushakov Ju. A. Issledovanie seti virtual'noj infrastruktury centra obrabotki dannyh s gibridnoj programmno-konfiguriruemoj kommutaciej [Research of virtual infrastructure network of data processing center with hybrid software-configurable switching]. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta* [Vestnik of Ryazan State Radio Engineering University]. 2021. No 75. Pp. 34-43. DOI: 10.21667/1995-4565-2021-75-34-43. (In Russ.)

10. Nikulchev E. V., Payain S. V., Pluzhnik E. V. Dinamicheskoe upravlenie trafikom programmno-konfiguriruemyh setej v oblachnoj infrastructure [Dynamic traffic control of cloud infrastructure with software-defined networking]. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta* [Vestnik of Ryazan State Radio Engineering University]. 2013. No 3 (45). P. 54-57. (In Russ.)

11. Leohin Ju. L., Fathulin T. D. Ocenka vozmozhnosti predostavlenija garantirovannoj skorosti peredachi dannyh v programmno-konfiguriruemoj opticheskoj seti [Estimation of the possibility to provide guaranteed data rate for a client in software-defined optical network]. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta* [Vestnik of Ryazan State Radio Engineering University]. 2020. No 71. P. 45-59. DOI: 10.21667/1995-4565-2020-71-45-59. (In Russ.)

12. Koryachko V. P., Perepelkin D. A., Ivanchikova M. A. et al. Programmnaja infrastruktura i vizual'naja sreda raspredelennoj obrabotki potokov dannyh v programmno-konfiguriruemyh setjah [Software infrastructure and visual environment of distributed data flows processing in software defined networks]. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta* [Vestnik of Ryazan State Radio Engineering University]. 2018. No 65. Pp. 44-54. DOI: 10.21667/1995-4565-2018-65-3-44-54. (In Russ.)

13. 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.

14. Kobayashi M., Seetharaman S., Parulkar G. et al. Maturing of OpenFlow and Software-Defined Networking Through Deployments. *Computer Networks*. 2014. Vol. 61. Pp. 151-175.

15. Nikishin K.I. Modelirovanie i verifikacija topologij programmno-konfiguriruemyh setej [Controller modeling and data transmission verification in software defined networks]. *Vestnik Rjazanskogo gosudarstvennogo radiotehnicheskogo universiteta* [Vestnik of Ryazan State Radio Engineering University]. 2022. No 80. Pp. 75-83. (In Russ.)

16. Nikishin K.I. Modelirovanie kontrollera i verifikacija processa peredachi dannyh v programmno-konfiguriruemyh setjah [Modeling and Verification of Software Defined Network Topology]. *Vestnik Rjazanskogo gosudarstvennogo radiotehnicheskogo universiteta* [Vestnik of Ryazan State Radio Engineering University]. 2022. № 80. Pp. 75-83. (In Russ.)

17. Wael Hosny Fouad Aly. A novel controller placement using Petri-nets for SDNs, *Wseas Transactions on Mathematics*. 2020. Vol. 19. Pp. 598-605.

18. Nikishin K.I. Modelirovanie besprovodnoj sensornoj seti s ispol'zovaniem OMNET++ [Modeling of wireless sensor network using OMNET++]. *Vestnik Rjazanskogo osudarstvennogo radiotehnicheskogo universiteta* [Vestnik of Ryazan State Radio Engineering University]. 2021. No 78. P. 46-54. (In Russ.)

19. Nikishin K.I, Konnov N.N. Generator trafika Ethernet na osnove cvetnyh setej Petri [The traffic generator of switch ethernet using colored Petri nets]. Modeli, sistemy, seti v jekonomike, tehnike, prirode i obshhestve [Models, systems, networks in economics, technology, nature and society]. 2016. No 1 (17). Pp. 299–307. (In Russ.).

The article was submitted 25.05.2022; approved after reviewing 10.06.2022; accepted for publication 15.06.2022

For citation: Nikishin K. I. Modeling of the Method of Early Diagnosis of Real-Time Traffic Losses in Software Defined Networks Based on the Apparatus of Petri Nets. *Vestnik of Volga State University of Technology. Ser.: Radio Engineering and Infocommunication Systems*. 2022. No 2 (54). Pp. 47–60. DOI: https://doi.org/10.25686/2306-2819.2022.2.47

Information about the author

Kirill I. Nikishin – Candidate of Engineering Sciences, senior lecturer of the sub-department of computer engineering of Penza State University. Research interests – networks; traffic transmission; telecommunication systems. The author of 47 scientific publications.

Author declare that they have no conflict of interest.
Author read and approved the final manuscript.