СБОРНИК **НАУЧНЫХ ТРУДОВ НГТУ. – 2019. – № 3–4 (96).** – 106–121

УДК 004.422.81:004.75:004.052.32 DOI: 10.17212/2307-6879-2019-3-4-106-121

# ИССЛЕДОВАНИЕ ОТКАЗОУСТОЙЧИВОСТИ ИНФРАСТРУКТУРЫ NSM*

В.В. КОМИССАРОВ[1], Д.О. ТИНГАЙКИН[2]

[1] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, магистрант кафедры вычислительной техники. E-mail: sodiz@yandex.ru*

[2] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, магистрант кафедры вычислительной техники. E-mail: teenguyking@gmail.com*

В работе будет рассмотрено программное обеспечение с открытым исходным кодом – проект Network Service Mesh, разрабатываемый компанией Cisco при содействии Red Hat, Huawei, Intel, Vmware. Данный проект расширяет сетевые возможности программного обеспечения Kubernetes, а именно позволяет динамически конфигурировать и создавать сетевые интерфейсы между несколькими POD. Классический случай установки соединения является случаем, когда клиентский и конечный POD находятся на одном узле. Также POD могут находиться на разных узлах.

В данной работе были проверены сценарии установки соединений между POD, находящимися на разных узлах, а также были проверены сценарии, когда один или несколько из ключевых компонентов инфраструктуры выходил из строя.

Под отказоустойчивостью в данной работе понимается, что система сохраняет работоспособность при отказе одного или нескольких основных компонентов инфраструктуры NSM. В инфраструктуру NSM входят следующие компоненты: сервисный менеджер (NSM), агент пересылки (Forwarder), а также вспомогательные контейнеры для сервисного менеджера, такие как клиент к реестру Kubernetes (nsmd-k8s), клиент для плагина устройств (nsmdp).

Исследование отказоустойчивости показало, что проект поддерживает 7 из 9 проверенных сценариев. Было обнаружено, что в данный момент отсутствует решение для проблемы, когда unix сокет файлы, используемые для коммуникации между компонентами, повреждаются и система не способна обнаружить эту неполадку, что приводит к ее неработоспособности. Также было обнаружено, что система не в состоянии оповестить клиента о падении в случае, когда все ключевые компоненты, кроме клиента и конечной точки, вышли из строя. Было предложено использовать мониторинг для решения данных проблем.

**Ключевые слова:** Kubernetes, DPAPI, сетевые интерфейсы, POD, сервисный менеджер L2/L3, отказоустойчивость, мониторинг, gRPC, процесс установления соединения

* Статья получена 30 августа 2019 г.

## СПИСОК СОКРАЩЕНИЙ

Application Service Mesh – реализация Service Mesh, в которой основное внимание уделяется нагрузкам L4 (TCP-потоки/UDP-дейтаграммы) и L7 (HTTP/протоколы, специфичные для приложений).

Network Service Mesh [1] – реализация Service Mesh, в которой основное внимание уделяется нагрузкам L2 (например, кадры) и L3 (например, пакеты).

Network Service – ориентированная на разработчика возможность для наблюдения за тем, как пакеты обрабатываются сетью.

Service Mesh – выделенный слой инфраструктуры для обеспечения взаимодействия между сервисами.

Network Service Manager (NSMgr) – это POD, развертываемый на каждый узел Kubernetes, который находится на уровне хоста, обеспечивая обработку сетевых запросов на уровне узла, а также формируя соединения с каждым другим NSMgr в домене NSR.

Определяющие характеристики Network Service Manager:

1) производит обработку вызовов gRPC и обеспечивает подключение к NSE.

2) регистрирует Network Services, предоставляемые Network Service Endpoints на уровне узла.

Forwarder – логическая конструкция, обеспечивающая сквозные соединения между NSC, NSE. Конфигурирует тип связи, механизмы и элементы пересылки к сетевому сервису.

Network Service Client – запросчик или потребитель сетевого сервиса.

Kubernetes [15] – открытое программное обеспечение для автоматизации развертывания, масштабирования контейнеризированных приложений и управления ими.

POD – базовая единица для управления и запуска приложений, один или несколько контейнеров, которым гарантирован запуск на одном узле, обеспечивающая разделение ресурсов.

Кластерная федерация [2] – специальный механизм, упрощающий управление множеством кластеров за счет синхронизации находящихся в них ресурсов и автоматического обнаружения сервисов во всех кластерах.

Admission Controller – нативная функция Kubernetes, которая помогает разработчикам определять и настраивать то, что разрешено запускать в кластере. Контроллер допуска перехватывает и обрабатывает запросы к API Kubernetes до сохранения объекта, но после проверки подлинности и авторизации запроса.

Request – в рамках данной статьи подразумевает запрос от Network Service Client с целью создания интерфейсов между клиентским и сервисным POD.

DPAPI (Kubernetes Device Plugin API) – Kubernetes предоставляет device plugin framework, который позволяет модифицировать спецификацию POD во время развертывания.

SR-IOV (Single Root Input/Output Virtualization) – технология виртуализации устройств, позволяющая предоставить виртуальным машинам прямой доступ к части аппаратных возможностей устройства.

## ВВЕДЕНИЕ

Проект NSM расширяет следующие сетевые возможности Kubernetes:

- соединения между несколькими POD как в рамках одного кластера, так и в рамках федераций;
- использование нестандартных протоколов;
- сетевой контекст как объект первого класса;
- управление сервисными функциями на основе политик (SFC);
- минимальная потребность в изменениях в Kubernetes;
- обеспечение динамических соединений по требованию.

Эти цели достигаются с помощью простого набора API-интерфейсов, предназначенных для облегчения подключения либо между контейнерами, на которых выполняются службы, либо с внешним NSM Endpoint. Новые соединения согласовывают свойство коммуникации, например:

- тип сетевого интерфейса (например, Linux Interface, MemIf или vhost-user);
- тип нагрузки (например, Ethernet, IP, MPLS или L2TP).

## 1. ПОСТАНОВКА ЗАДАЧИ

- Рассмотреть сценарии установки соединений между POD, находящимися на разных узлах, при помощи NSM.
- Рассмотреть сценарии, когда один или несколько из ключевых компонентов инфраструктуры NSM выходит из строя.

## 2. ОСНОВНАЯ ЧАСТЬ

### Пример запроса на создание сетевых интерфейсов

Примером NSM является запрос доступа к внешнему интерфейсу. Рассмотрим Kubernetes POD, который делает запрос на сетевой интерфейс. Интерфейс сконфигурирован и подключен в POD. Этот сценарий может проис-

ходить либо посредством динамического вызова от процесса в POD, либо из init-контейнера [12], который настраивает подключение перед основным POD-контейнером.

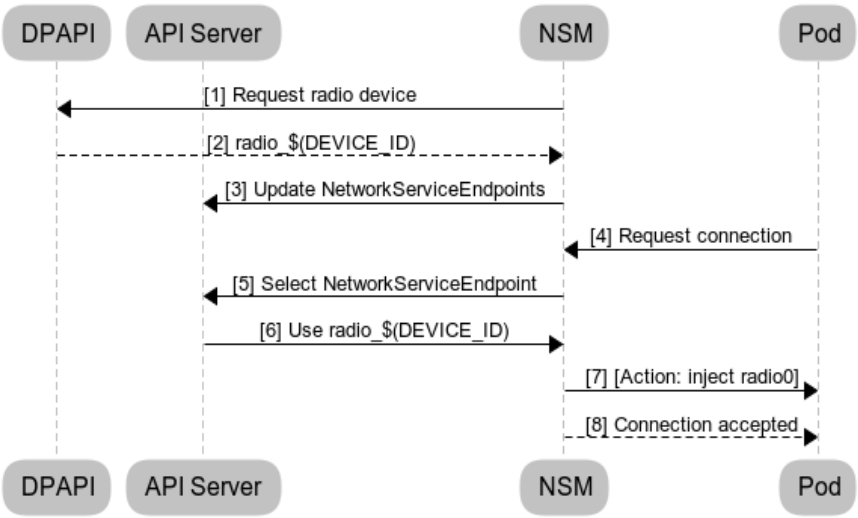


*Рис. 1.* Схема Request-запроса

**Добавление сервиса**

NSM также взаимодействует с DPAPI. В этом сценарии мы взаимодействуем со службой с ограниченным количеством интерфейсов. Другой вариант использования с похожим шаблоном – виртуализация ввода-вывода с единым корнем (SR-IOV).

**NSM как решение сетевых проблем нового поколения**

Многогранные сети [8], такие как сети телекоммуникационных компаний, интернет-провайдеров и передовые корпоративные сети, реорганизуют свои решения с появлением ряда новых сетевых технологий, включая следующее:

- виртуализацию сетевых функций (NFV);
- сети 5G;
- граничные вычисления;
- устройства интернета вещей.

Каждая из этих технологий обеспечивает значительное увеличение общего количества подключенных устройств, доступной пропускной способности на устройство и нагрузки на облачные сервисы [7].

Операторы многогранных сетей с расширенными вариантами использования L2/L3 в настоящее время находят решения [10], основанные на контейнеризации, плохо подходящими для их архитектуры следующего поколения. Отсутствие поддержки расширенных сценариев использования в сети активно препятствует принятию новой облачной парадигмы несколькими отраслями [3].

Технологии создания контейнерных сетей текущего поколения в первую очередь ориентированы на однородные кластеры приложений, на предприятия с сетями с малой задержкой и высокой пропускной способностью [4]. Эти предположения не соответствуют потребностям телекоммуникационных компаний, интернет-провайдеров и современных корпоративных сетей [5].

Наконец, существующие нативные облачные решения допускают динамическое конфигурирование развертывания [14], но реализация этого развертывания в основном неизменна. Container Networking Interface (CNI) занимается только распределением сети на этапах инициализации и удаления POD'а.

Текущий шаблон работает для соединения монолитных виртуальных сетевых функций (VNFs) вместе, но противоположен нативной парадигме облака. Простое преобразование существующих монолитных шаблонов VNF в собственные эквиваленты в облаке по своей сути не обеспечивает масштабируемость, отказоустойчивость и управляемость, которых ожидают пользователи – держатели облака [16]. Фактически перевод текущих моделей может привести к повышению общей стоимости владения из-за затрат на рефакторинг инфраструктуры, тестирование и обучение, одновременно уменьшая изоляцию инфраструктуры и создавая простор для разного рода атак [11].

Чтобы реализовать все преимущества нативной облачной парадигмы для сценариев использования NFV, необходимо выполнить как минимум два условия:

- VNF должны быть изменены или переписаны для отражения архитектуры, найденной в 12-факторных приложениях;
- должен существовать API, который позволяет VNF определять свое сетевое намерение динамически и через абстрактный API.

Когда эти два условия соблюдены, приложения NFV могут быть горизонтально масштабированы при эффективном использовании сетевых ресурсов. Когда задействована программно-определяемая сеть (SDN), разработка приложений NFV и SDN также может происходить независимо от слабой связи.

**Исследование отказоустойчивости**

Далее мы рассмотрим, как проект Network Service Mesh реагирует на события и ситуации, возникающие на разных этапах жизненного цикла, а также рассмотрим улучшения в этой области.

В терминологии NSM реакция на любые ситуации с клиентским соединением называется «Healing» соединением, поэтому Network Service Manager постарается как можно скорее «излечить» соединение с помощью наиболее подходящего способа, чтобы восстановить сетевую инфраструктуру для клиента [13].

Давайте рассмотрим следующий сетевой сценарий: у нас есть три узла кластера, две конечные точки и один клиент. Состояние после установления соединения отображено на рис. 2.

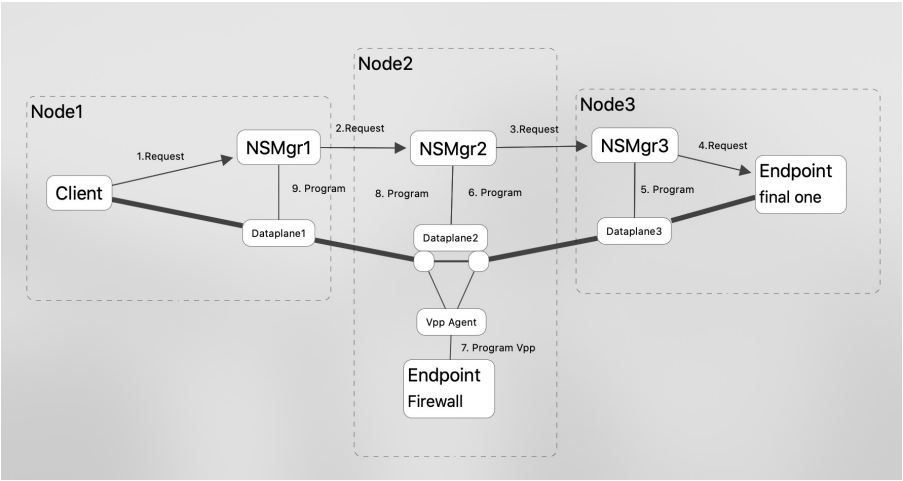


*Рис. 2*. Проверяемый случай – 3 узла

Рассмотрим подробнее, что происходит на каждом этапе.

Node1:

- Клиент запрашивает у локального NSMgr сервис, и соединение устанавливается с использованием промежуточного сервиса под названием «Endpoint/Firewall».

Node2:

- Firewall Endpoint выполняет фильтрацию и проверку соединения и передает его к Final Endpoint.

Node3:

- Final Endpoint принимает соединение, назначает IP-адрес, предоставляет ресурсы и передает его обратно. После этого у Forwarder находящийся на Node1 конфигурирует сетевые интерфейсы для client POD и для Final Endpoint.

Рассмотрим сценарии отказоустойчивости согласно рис. 2.

Далее мы провели ряд экспериментов и обнаружили поддерживаемые и неподдерживаемые сценарии отказоустойчивости.

Таблица 1

**Поддерживаемые сценарии отказоустойчивости**

| № | Сценарий | Обработка сценария |
|---|---|---|
| 1 | NSMgr1 падает unix socket NSE/Client был создан | NSMgr2 обнаруживает сбой и помечает соединение как Healing на удаленной стороне.<br>Перезагрузка NSMgr1:<br>1) чтение файла clients/NSE,<br>2) пинг восстановленного списка NSE,<br>3) получение списка соединений от Forwarder,<br>4) восстановление с помощью NSMgr2 и т. д. |
| 2 | Forwarder1/2/3 падает | 1. NSMgr1/2/3 обнаруживает, что Forwarder падает, ожидает новый Forwarder.<br>2. Перепрограммирование нового локального Forwarder с информацией о соединении, которую мы имеем.<br>3. Передача обновленного соединения мониторам.<br>4. Каждый NSMgr при получении соединения обновляется, перепрограммирует локальный Forwarder и отправляет обновление |
| 3 | Endpoint Firewall падает | 1. NSMgr2 обнаруживает падение NSE в процессе мониторинга. Восстановление будет продолжено на NSMgr1.<br>2. NSMgr1:<br>• удаленно Peer обнаруживает, что часть DST соединения падает;<br>• начало ожидания доступности другого NSE с таким же сервисом и выполнение запроса на него;<br>• запрос нового NSE и подготовка полной цепочки, оценка Final Endpoint в качестве нового клиента и т. д.;<br>• Heal connection.<br>3. NSMgr3 с помощью Forwarder обнаруживает, что клиент падает, и закрывает соединения |

Окончание табл. 1

| № | Сценарий | Обработка сценария |
|---|---|---|
| 4 | Final Endpoint падает | 1. NSMgr3 обнаруживает, что NSE падает. Восстановление будет продолжено на NSMgr2.<br>2. NSMgr2:<br>• ожидание восстановления NSE;<br>• запрос NSE, или отправка «соединение прервано», если время ожидания NSE отсутствует;<br>• Forwarder обновляет состояние.<br>3. NSMgr1 получает обновление и перепрограммирует Forwarder, отправляет обновление на клиентский монитор |
| 5 | Node3 падает | NSMgr2 обнаруживает соединения, ждет таймаут от того же NSE, если не появляется, выбирает другой для выполнения соединения |
| 6 | Node2 падает и перезапускается | NSMgr1 обнаруживает падение удаленного соединения и ждет его повторного появления, если будет выполнено восстановление на том же NSMgr2.<br>NSMgr3 обнаруживает разрыв соединения с источником и закрывает соединение с NSE |
| 7 | Node2 падает и не перезапускается | Если Node2 полностью сломан, NSMgr3 обнаруживает, что локальное соединение упало, и закрывает соединение NSE.<br>NSMgr1 ждет появления NSMgr2 в течение тайм-аута, и если этого не произойдет, то запрашивает ту же сетевую службу для любого другого доступного NSMgr (например, NSMgr3) с этой службой и выполняет восстановление соединения |

Таблица 2

**Неподдерживаемые сценарии отказоустойчивости**

| № | Сценарий | Обработка сценария |
|---|---|---|
| 1 | NSMgr1 падает<br>Нет unix сокета<br>NSE/Client или<br>он поврежден | NSMgr2 обнаруживает сбой и помечает соединение как Healing на удаленной стороне.<br>Перезагрузка NSMgr1:<br>1) попытка найти unix socket clients/NSE -> ОШИБКА;<br>2) получение списка соединений от Forwarder.<br>Для удаленного NSE будет выполнено восстановление.<br>Для локального NSE они не будут найдены и будут помечены как закрытые |
| 2 | Оба NSMgr<br>и Forwarder падают<br>и перезапускаются | NSMgr пропингует все NSE и перерегистрирует их. Поскольку Forwarder упал, никакие соединения не могут быть восстановлены.<br>NSMgr2 обнаружит разрыв соединения с источником и закроет локальное соединение и окончательное соединение NSE.<br>Монитор на стороне клиента. Сможет найти NSMgr для повторного подключения через сокет Unix или TCP и снова запросит соединение |

1. Network Service Mesh должен обеспечивать возможность восстановления из наиболее подходящих ситуаций, таких как падение узлов, перезапуск служб, завершение работы конечных точек, а также позволять менять ситуации, и менять их плавно. Поэтому он должен восстановить запрашиваемую клиентом сетевую инфраструктуру в короткие сроки и уведомить клиентов, что всё в порядке.

2. Вторым важным моментом является установление соединения и восстановление. Оба по своей природе одинаковы при передаче запроса конечным точкам и администратору сетевых служб и в данный момент программируют развернутые Forwarder, запрашивая синхронный доступ к сетевому сервису, а восстановление асинхронное, поэтому нужно наладить их совместную работу.

3. Прямо сейчас NSM не знает, установит ли Endpoint больше соединений через NSM и сделает цепочку запросов, но для правильного восстановления конечной точки мы должны быть уверены и ждать поступления запроса на восстановление для плавного восстановления клиентского соединения.

**Предлагаемые решения**

1. Улучшение процесса установления соединения с использованием непрерывного мониторинга.

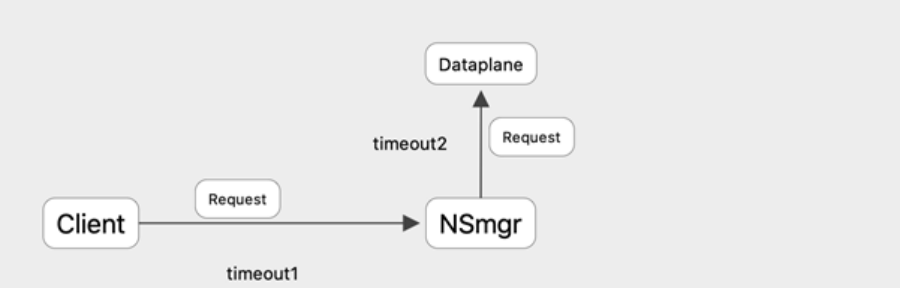


*Рис. 3.* Процесс установления соединения на данный момент

Прямо сейчас NSM использует синхронный API в запросе начального соединения.

И после того, как соединение установлено, мы будем вызывать монитор для прослушивания обновлений соединения. Монитор будет являться существенной частью и работать в любом сценарии.

Таким образом, предложение состоит в том, чтобы перейти от синхронной операции для запроса соединения как на NSM/Remote NSM, так и на Forwarder, чтобы быть асинхронным (рис. 4).

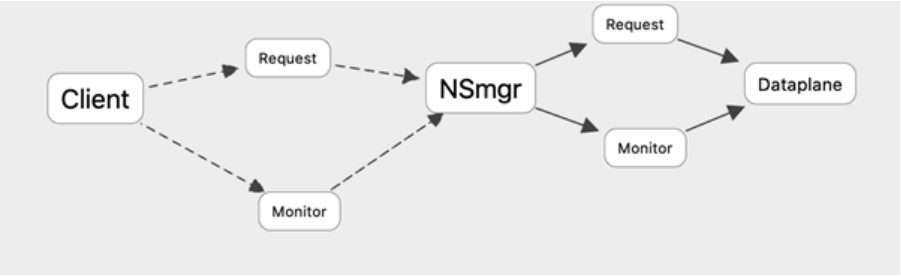


*Рис. 4.* Предлагаемый процесс соединения

Это решит проблему с тайм-аутами контекста при передаче выполнения от одной части NSM к другой, и поскольку у нас будут мониторы в каждой точке, это будет одинаково для восстановления и запроса соединения с точки зрения клиента.

Также монитор будет отвечать за отслеживание времени ожидания соединения на каждом шаге и информировать клиентов.

2. Улучшение восстановления с использованием мониторинга.

Для этого нам необходим монитор подключений, который будет неотъемлемой частью как клиентов на базе SDK, так и клиентов с Admission Controller/плагином устройства/ручными настройками.

В обоих случаях сразу после установления соединения он начнет мониторинг состояния соединения, а в случае разрыва соединения NSMgr выполнит повторное подключение к нему и переустановит соединение.

И если мы перейдем к асинхронным API-интерфейсам, монитор будет использоваться для отслеживания состояния соединения.

Это позволит восстановиться в любой ситуации, так как если мы отбросим и восстановим всю инфраструктуру, а клиент останется нетронутым, все необходимое рабочее состояние будет восстановлено.

В случае SDK он будет частью цепочки установки, и это должно позволить более плавно обрабатывать эти ситуации, информируя код пользователя о подключении, находящемся сейчас в состоянии восстановления, и информируя обо всех произошедших событиях.

Таким образом, компонент мониторинга или вспомогательный модуль мониторинга отвечает за вызов запроса на то же соединение, полученного от NSMgr в случае потери соединения с ним.

2.1. Восстановление на стороне клиента.

Прямо сейчас клиент делает запрос впервые, и если NSMgr полностью упадет, то клиент об этом не узнает, что может привести к нежелательным последствиям.

С подходом мониторинга можно избежать этой ситуации (см. табл. 2). Монитор сообщит клиенту об обрыве соединения, и клиент примет решение, восстанавливать соединение или нет.

В этом случае клиент может снова выполнить Request с некоторыми новыми параметрами или попытаться выполнить восстановление, это будет то же самое с точки зрения NSMgr.

3. Модификация протокола соединения.

В многоцепных соединениях NSE, когда запрашиваемый NSE может являться клиентом к NSE’ (рис. 5), нам нужно предоставить возможность указать любое количество контекстов соединения индивидуально для NSM и NSE, которые мы передаем.

Прямо сейчас есть только один контекст, заполненный NSE и переданный обратно соединению, на каждом шаге обновляются / заменяются значения конфигураций соединения.

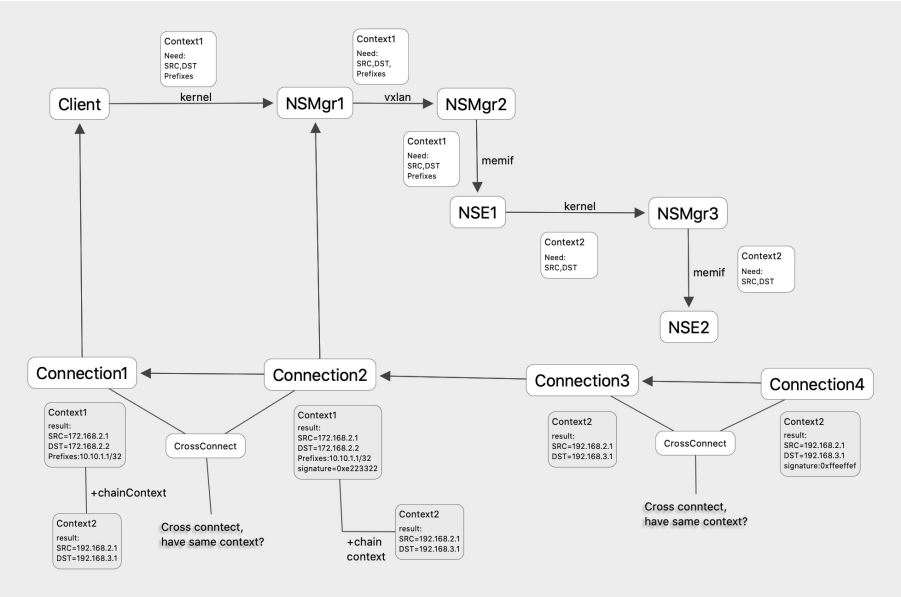


*Рис. 5.* Протоколы установки соединения

В случае падения одного из промежуточных NSMgr, например NSMgr2 (рис. 5), произойдет потеря данных (Connection 3), которые были заполнены на стороне NSMgr2.

В случае восстановления соединения эта информация может быть использована для выбора правильного соединения и правильной конечной NSE, к которой должен подключиться NSC.

Мы предлагаем использовать список соединений вместо использования одного экземпляра данных, содержащих информацию о соединении.

Список может быть создан с использованием служебного открытого ключа подписи RSA [9], чтобы избежать таких сетевых угроз, как «атака посредника» [6].

## БЛАГОДАРНОСТЬ

Авторы выражают искреннюю благодарность профессору кафедры автоматики А.А. Воеводе за помощь при выполнении работ, а также полезное обсуждение полученных результатов.

## ЗАКЛЮЧЕНИЕ

Рассмотрена архитектура проекта Network Service Mesh, рассмотрен случай сетевого конфигурирования на трех узлах с тестированием нескольких сценариев отказоустойчивости (см. табл. 1).

Найдены проблемы в существующих механизмах отказоустойчивости (см. табл. 2), а именно: отсутствие решения для обработки сценария повреждения служебных unix сокет файлов после перезапуска NSMgr. Данную проблему мы предлагаем решить, расширяя протокол соединения. Также была найдена проблема отсутствия уведомления клиента о падениях NSMgr и Forwarder, и нами было предложено использовать мониторинг на стороне NSC.

## СПИСОК ЛИТЕРАТУРЫ

1. Network Service Mesh. The Hybrid/Multi-cloud IP Service Mesh: website. – URL: https://networkservicemesh.io/ (accessed: 18.12.2019).

2. *Лукша М.* Kubernetes в действии. – М.: ДМК Пресс, 2018. – 672 с.

3. *Стивенс У.Р., Раго С.А.* UNIX. Профессиональное программирование. – 3-е изд. – СПб.: Питер, 2018. – 944 с.

4. *Зелигер Н.Б., Чугреев О.С., Яновский Г.Г.* Проектирование сетей и систем передачи дискретных сообщений. – М.: Радио и связь, 2015. – 176 с.

5. *Кульгин М.* Технологии корпоративных сетей. – СПб.: Питер, 2019. – 704 с.

6. *Рэтлифф Б., Баллард Д.* Microsoft Internet security and acceleration (ISA) server 2004: справочник администратора. – М.: Русская редакция, 2017. – 400 с.

7. *Спортак М.А., Паппас Ф.Ч., Рензинг Э.* Компьютерные сети. Кн. 1. High-Perfomance Networking. – М.: ДиаСофт, 2016. – 432 с.

8. *Столлингс В.* Современные компьютерные сети. – СПб.: Питер, 2017. – 783 с.

9. *Цвики Э., Купер С., Чапмен Б.* Создание защиты в интернете. – 2-е изд. – М.; СПб.: Символ-Плюс, 2019. – 928 с.

10. *Щербо В.К., Киреичев В.М., Самойленко С.И.* Стандарты по локальным вычислительным сетям. – М.: Радио и связь, 2015. – 304 с.

11. *Applegate I.* How process isolation became viable for production deployment. – URL: http://containerjournal.com/2016/02/11/how-process-isolation-became-viable-for-production-deployment/ (accessed: 18.12.2019)

12. *Grant B.* Kubernetes design and architecture. – URL: https://github.com/kubernetes/community/blob/master/contributors/design-proposals/architecture/architecture.md (accessed: 18.12.2019).

13. Open Container Initiative. Runtime Specification. – URL: https://github.com/opencontainers/runtime-spec (accessed: 18.12.2019).

14. *Turnbull J.* The Docker Book. – URL: https://dockerbook.com/ (accessed: 18.12.2019).

15. Официальная документация по Kubernetes. – URL: https://kubernetes.io/ (дата обращения: 18.12.2019).

16. Open Container Initiative. Image Specification. – URL: https://github.com/opencontainers/image-spec (accessed: 18.12.2019).

***Комиссаров Валерий Владимирович***, магистрант кафедры вычислительной техники Новосибирского государственного технического университета по направлению «Информатика и вычислительная техника». Имеет одну публикацию. E-mail: sodiz@yandex.ru

***Тингайкин Денис Олегович***, магистрант кафедры вычислительной техники Новосибирского государственного технического университета по направлению «Информатика и вычислительная техника». Имеет две публикации. E-mail: teenguyking@gmail.com

DOI: 10.17212/2307-6879-2019-3-4-106-121

## Investigation of the fault tolerance of the NSM infrastructure*

**V.V. Komissarov[1], D.O. Tingajkin[2]**

[1] *Novosibirsk State Technical University, 20 Karl Marx Prospekt, Novosibirsk, 630073, Russian Federation, master student of the department of computer engineering. E-mail: sodiz@yandex.ru*

[2] *Novosibirsk State Technical University, 20 Karl Marx Prospekt, Novosibirsk, 630073, Russian Federation, master student of the department of computer engineering. E-mail: teenguyking@gmail.com*

The work will consider open source software - the Network Service Mesh project developed by Cisco, with the assistance of Red Hat, Huawei, Intel, Vmware. This project extends the networking capabilities of Kubernetes software, namely it allows you to dynamically configure and create network interfaces between multiple PODs. The classic connection setup case is when the client and destination PODs are on the same node. PODs can also be on different nodes.

In this paper, we tested the scenarios for establishing connections between PODs located on different nodes, and also tested the scenarios when one or more of the key infrastructure components failed.

* *Received 30 August 2019.*

In this work, fault tolerance is understood to mean that the system remains operational in the event of failure of one or more of the main NSM infrastructure components. The NSM infrastructure includes the following components: service manager (NSM), forwarding agent (Forwarder), as well as auxiliary containers for the service manager such as: client to the Kubernetes registry (nsmd-k8s), client for the device plug-in (nsmdp).
Resiliency research has shown that the project supports 7 out of 9 proven scenarios. It was found that at the moment there is no solution for the problem when the unix socket files used for communication between components are damaged and the system is unable to detect this problem, which leads to its inoperability. It was also found that the system is not able to notify the client of a crash when all the key components except the client and the endpoint have failed. It was suggested that monitoring be used to solve these problems.

**Keywords:** Kubernetes, DPAPI, network interfaces, POD, L2/L3 service manager, fault tolerance, monitoring, gRPC, connection establishment process

## REFERENCES

1. *Network Service Mesh. The Hybrid/Multi-cloud IP Service Mesh*: website. Available at: https://networkservicemesh.io/ (accessed 18.12.2019).

2. Lukša M. *Kubernetes in action*. Shelter Island, NY, Manning Publications Co., 2018 (Russ. ed.: Luksha M. *Kubernetes v deistvii*. Moscow, DMK Press Publ., 2018. 672 p.).

3. Stevens W.R., Rago S.A. *Advanced programming in the UNIX environment*. 3rd ed. Upper Saddle River, NJ, Addison-Wesley, 2013 (Russ. ed.: Stivens U.R., Rago S.A. *UNIX. Professional'noe programmirovanie*. 3rd ed. St. Petersburg, Piter Publ., 2018. 944 p.).

4. Zeliger N.B., Chugreev O.S., Yanovskii G.G. *Proektirovanie setei i sistem peredachi diskretnykh soobshchenii* [Design of networks and systems for the transmission of discrete messages]. Moscow, Radio i svyaz' Publ., 2015. 176 p.

5. Kul'gin M. *Tekhnologii korporativnykh setei* [Technologies of corporate networks]. St. Petersburg, Piter Publ., 2019. 704 p.

6. Ratliff B., Ballard J. *Microsoft Internet security and acceleration (ISA) server 2004*: *administrator's pocket consultant*. Redmond, Microsoft Press, 2006 (Russ. ed.: Ratliff B., Ballard D. *Microsoft Internet security and acceleration (ISA) server 2004*: *spravochnik administratora*. Moscow, Russkaya redaktsiya Publ., 2017. 400 p.).

7. Sportak M.A., Pappas F.Ch., Renzing E. *Komp'yuternye seti*. Kn. 1. *High-Perfomance Networking* [Computer networks. Bk. 1. High-Performance Networking]. Moscow, DiaSoft Publ., 2016. 432 p. (In Russian).

8. Stallings W. *High-speed networks and internets: performance and quality of service*. Upper Saddle River, Prentice Hall, 2002 (Russ. ed.: Stollings V. *Sovremennye komp'yuternye seti*. St. Petersburg, Piter Publ., 2017. 783 p.).

9. Zwicky E., Cooper S., Chapman B. *Building Internet Firewalls*. 2nd ed. Beijing, Cambridge, O'Reilly, 2000 (Russ. ed.: Tsviki E., Kuper S., Chapmen B. *Sozdanie zashchity v internete*. 2nd ed. Moscow, St. Petersburg, Simvol-Plyus Publ., 2019. 928 p.).

10. Shcherbo V.K., Kireichev V.M., Samoilenko S.I. *Standarty po lokal'nym vychis-litel'nym setyam* [Standards for local area networks]. Moscow, Radio i svyaz' Publ., 2015. 304 p.

11. Applegate I. How process isolation became viable for production deployment. Available at: http://containerjournal.com/2016/02/11/how-process-isolation-became-viable-for-production-deployment/ (accessed 18.12.2019).

12. Grant B. *Kubernetes design and architecture*. Available at: https://github.com/kubernetes/community/blob/master/contributors/design-proposals/architecture/architecture.md (accessed 18.12.2019).

13. Open Container Initiative. Runtime Specification: website. Available at: https://github.com/opencontainers/runtime-spec (accessed 18.12.2019).

14. Turnbull J. *The Docker Book*. Available at: https://dockerbook.com/ (accessed 18.12.2019).

15. Kubernetes official documentation. Available at: https://kubernetes.io/ (accessed 18.12.2019).

16. Open Container Initiative. Image Specification. Available at: https://github.com/opencontainers/image-spec (accessed 18.12.2019).

Для цитирования:

*Комиссаров В.В., Тингайкин Д.О.* Исследование отказоустойчивости инфраструктуры NSM // Сборник научных трудов НГТУ. – 2019. – № 3–4 (96). – С. 106–121. DOI: 10.17212/2307-6879-2019-3-4-106-121.

For citation:

Komissarov V.V., Tingajkin D.O. Issledovanie otkazoustoichivosti infrastruktury NSM [Investigation of the fault tolerance of the NSM infrastructure]. *Sbornik nauchnykh trudov Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta – Transaction of scientific papers of the Novosibirsk state technical university*, 2019, no. 3–4 (96), pp. 106–121. DOI: 10.17212/2307-6879-2019-3-4-106-121.