УДК 004.75

ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ SERVICE MESH МОДУЛЯ

[1, 2]Тазиева Р.Ф., [2]Титов А.Н.

[1]ФГБОУ ВО «Казанский государственный энергетический университет», Казань, e-mail: ram89_89@mail.ru;

[2]ФГБОУ ВО «Казанский национальный исследовательский технологический университет», Казань, e-mail: redposition53@mail.ru

Приведено описание Service Mesh модуля для организации взаимодействия между агентами системы мониторинга. Программный модуль обладает следующей функциональностью: регистрацией устройств в системе с определением каждого экземпляра устройства, его типа, текущего адреса; организацией маршрутизации между клиентами сети; возможностью интеграции с существующими сервисами посредством HTTP-проксирования; возможностью динамического изменения текущей зоны работы. В разработанном программном модуле использованы известные архитектурные подходы: Solid, CQRS, луковая архитектура. Реализована архитектура Service Discovery сервера, Sidecar Proxy. Приведена реализация HTTP proxy, описаны основные используемые в системе сущности. Программный интерфейс системы представлен в виде RESTful api соответствующей спецификации OpenApi с использованием инструмента Swagger. Алгоритм определения текущей зоны работы использует географическое или сетевое местоположение. Процесс построения карты доступности – ориентированный граф, представленный в виде матрицы смежности. Для нахождения кратчайшего пути между адресатом и отправителем использован алгоритм Дейкстры. На примере работы системы с определенной топологией и заданными правилами маршрутизации показано изменение поведения системы при перемещении сервиса, выступающего в качестве подвижного элемента, из одной зоны в другую.

Ключевые слова: системы мониторинга, Service Mesh модуль, маршрутизация, топология, запрос

PECULIARITIES OF THE SERVICE MESH DEVELOPMENT

[1, 2]Tazieva R.F., [2]Titov A.N.

[1]Kazan State Power Engineering University, Kazan, e-mail: ram89_89@mail.ru;

[2]Kazan National Research Technological University, Kazan, e-mail: redposition53@mail.ru

The Service Mesh description for organization of monitoring system agents' interaction is given. The software has the following functionality: devices logging option in the system, with the definition of each instance, its type, current address; routing option between network clients; integration option with existing services through HTTP-proxying; current working area dynamical changing option. In developed software general architectural approaches such as Solid, CQRS, onion architecture were used. The Service Discovery server and Sidecar Proxy architectures were carried out. HTTP proxy realization is given, the main entities used in the system are described. The system software interface presented in the form of a RESTful api corresponding to the OpenApi specification by using the Swagger tool. The current working area determination algorithm uses geographic or network location. The process of accessibility map building corresponds to an orgraph represented as an adjacency matrix. Dijkstra's algorithm for determination of the shortest path between the destination station and the sender was used. For example, a system with a certain topology and specified routing rules is taken. The system behavior changing in condition of tool-service (acting as a moving system element from one area to another) moving is shown.

Keywords: monitoring systems, Service Mesh, routing, topology, query

В ходе работы любой, даже самой качественной, системы могут возникать непредвиденные ситуации, выводящие систему из состояния равновесия. Самое важное в данном случае – вовремя отреагировать и провести восстановительные мероприятия. Для решения данной проблемы применяют системы мониторинга.

С увеличением размера информационных систем вопрос организации взаимодействия между компонентами системы становится более актуальным. С появлением таких концепций, как «Интернет вещей», и всеобщим увеличением количества устройств в сети важной становится простота организации работы территориально распределенных систем, включающих в себя эти устройства.

Примером таких систем являются корпоративные информационные системы. Для успешной организации процесса мониторинга распределенных корпоративных систем необходимы системы мониторинга, способные работать в условиях сложной топологии сети, учитывающие сложные бизнес-процессы компаний. При этом нужно иметь в виду специфику процесса мониторинга распределенных корпоративных систем:

1) сложность сетевой топологии и маршрутизации;

2) жесткое разграничение материальной ответственности. Каждый физический и электронный ресурс принадлежит определенной структурной единице компании;

3) применение принципов делегирования. Для обеспечения непрерывного функ-

ционирования бизнеса работы по ремонту оборудования могут быть переданы другой структурной единице;

4) наличие подвижных элементов системы.

Учитывая вышеизложенное, для решения этих проблем необходимо определить способ организации сетевого взаимодействия, чтобы обеспечить следующую функциональность системы:

1) определение зоны ответственности;

2) возможность динамической маршрутизации данных согласно заданным правилам;

3) возможность динамического подключения новых элементов сети без предварительной конфигурации.

Существующие Service Mesh решения [1] не способны выполнить поставленную задачу из-за ориентации существующего ПО на облачные решения, подход демонстрирует незрелость инструментов при использовании в территориально-распределенных системах. В существующих решениях отсутствует механизм зонирования по геолокации. Разработка системы, решающей эти проблемы, позволит организовать процесс мониторинга в распределенных корпоративных системах.

В качестве объекта исследования выступают информационные процессы мониторинга распределенных корпоративных систем. Предмет исследования – Service Mesh модуль системы мониторинга, обеспечивающий взаимодействие элементов системы.

Цель исследования – проектирование и разработка прототипа Service Mesh системы с учетом специфики процессов мониторинга распределенных корпоративных систем.

Методы исследования и выбранные архитектурные программные решения

Теоретико-методологическими основами на этапе проектирования Service Mesh модуля послужили теоретические исследования распределенных и корпоративных систем [2, 3] и систем мониторинга IT инфраструктуры [4, 5].

Проектируемый программный модуль должен обладать следующей функциональностью: регистрацией устройств в системе с определением каждого экземпляра, его типа, текущего адреса; организацией маршрутизации между клиентами сети; возможностью интеграции с существующими сервисами посредством HTTP-проксирования; возможностью динамического изменения текущей зоны работы, необходимой для работы внутри корпоративной информационной системы.

В качестве технологического стека для разработки Service Mesh модуля выбрана платформа .NET от компании Microsoft. Основные архитектурные подходы – Solid, CQRS, луковая архитектура; программная архитектура Service Discovery сервера и Sidecar Proxy; реализация HTTP proxy.

Основные составляющие, используемые в системе: программный интерфейс системы, соответствующий спецификации OpenAPI; алгоритм определения текущей зоны работы, использующий географическое или сетевое местоположение; процесс инициализации системы, особенности, необходимый для работы минимум конфигурации; процесс построения карты доступности – ориентированный граф, заданный в виде матрицы смежности; алгоритм построения маршрута – алгоритм Дейкстры для нахождения кратчайшего пути между адресатом и отправителем.

На рис. 1 изображена структура решения. Каждый проект соответствует определенному слою.

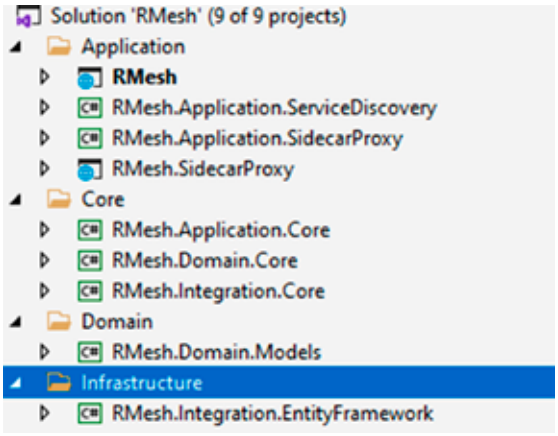

Рис. 1. Структура решения в Visual Studio

Уровень приложения и уровень домена зависят от ядра. Уровень инфраструктуры и клиентский уровень зависят от уровня приложений. Связь между инфраструктурой и веб- частью не нарушает установленные принципы, так как они находятся в одном слое.

На рис. 2 изображена диаграмма классов Sidecar Proxy. За логику работы приложения отвечают следующие классы:

1) SidecarManager является точкой входа в процесс работы приложения. Экземпляр данного класса создается во время запуска Sidecar Proxy. Основная задача – управление верхнеуровневой логикой приложения;

2) AreaProcessor содержит в себе логику определения текущего местоположения Sidecar Proxy;

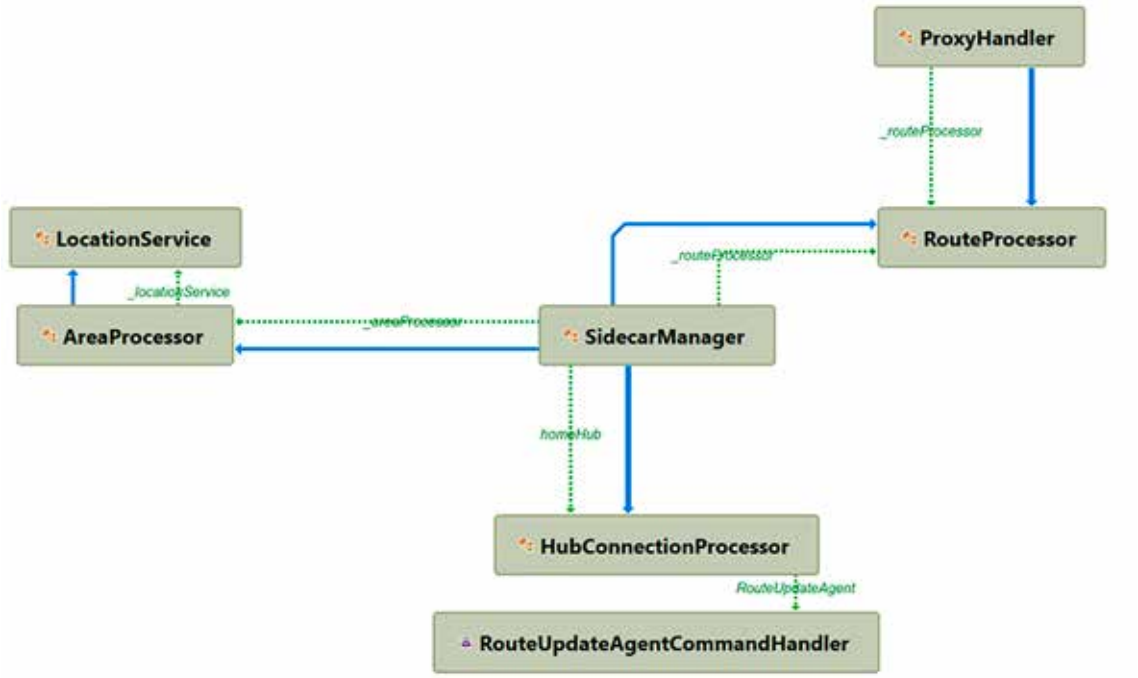

Рис. 2. Диаграмма классов Sidecar Proxy

3) Location service содержит в себе логику запроса геолокации;

4) HubConnection processor управляет соединением по протоколу WebSocket с сервером;

5) RouteProcessor отвечает за обработку и создание карты маршрутов к другим Sidecar Proxy;

6) ProxyHandler отвечает за логику перенаправления запросов.

Фактически Sidecar Proxy – это такое же Asp.net Core приложение, как и Service Discovery, главное отличие в том, что Sidecar Proxy не предоставляет API и не работает с базой данных. Основная функция системы – перенаправление запросов согласно заданным правилам. Для выполнения данного требования Sidecar Proxy и Service Discovery имеют в своем составе http proxy.

Согласно требованиям, необходимо хранить и обрабатывать данные о подключенных Sidecar Proxy, Service Discovery серверах, зонах ответственности и настройки маршрутизации. Основные сущности БД, используемые в работе системы:

1) агент – сущность, используемая для хранения данных о подключенных к системе Sidecar Proxy;

2) тип агента – сущность, которая описывает уникальный тип приложения, которое работает рядом с Sidecar Proxy. Например, сервер системы мониторинга, агент системы мониторинга;

3) маршруты типа агента – сущность, определяющая логику маршрутизации внутри системы;

4) зона – сущность определяющая зону действия определенного филиала компании;

5) Service Discovery – сущность, которая содержит информацию о существующих Service Discovery серверах.

Первый этап заключается в определении зон ответственности разных филиалов компании. Система поддерживает два способа определения границ: географический – для определения зоны используются географические координаты; сетевой – для определения зоны используются сетевой адрес Service Discovery сервера и маски сетей, определяющие внутреннюю сетевую структуру. В процессе работы программы связь поддерживается с двумя Service Discovery серверами: из домашней и текущей зоны.

Для описания географического местоположения зоны используется набор географических точек в формате широта, долгота. В процессе работы Sidecar Proxy определяет свое местоположение с помощью встроенной в базовую библиотеку .Net абстракции – CoordinateWatcher. Если на устройстве недоступен GPS, приложение запросит сетевой адрес с помощью открытого API ipstack.com. После получения своего местоположения Sidecar Proxy геометрически определяет вхождение точ-

ки в границы каждой из зон, тем самым определяя текущую зону. Следует учитывать то, что определение местоположения по сети является неточным. Погрешность определения может достигать сотен километров в случае, если сотовый оператор использует сеть из другого города.

Второй способ определения текущей зоны – сетевой способ. В каждой зоне расположен один Service Discovery сервер, для которого определены внутренний адрес и маски сетей, описывающие внутреннюю структуру. В процессе определения сетевого местоположения приложения осуществляют трассировку по заданному внутреннему адресу.

Любое серверное внутреннее ПО всегда находится внутри домашней зоны, оно неподвижно по определению. Для определения местоположения существует специальный параметр – AreaSetupType. AreaSetupType.Static означает, что Sidecar Proxy статичен и не выходит за пределы одной зоны, или дополнительная логика маршрутизации не имеет смысла. AreaSetupType. Dynamic означает, что Sidecar Proxy способен перемещаться, например Sidecar Proxy, установленный на корпоративный ноутбук сотрудника компании. Во время работы системы возможны ситуации, когда результаты, полученные двумя способами, будут противоречить друг другу: Sidecar находится в географической зоне одного филиала, но при этом подключен к внутренней сети компании через VPN, или в компании у разных филиалов совпадает внутренняя сетевая топология. Для определения порядка проверки указанными способами используется параметр AreaSetupPriority.

Для работы системы необходимо точно определять, какие Sidecar Proxy и Service Discovery серверы способны взаимодействовать друг с другом. В условиях работы внутри сложной распределенной корпоративной информационной сети возможны ситуации, когда два приложения не способны связаться. Для решения этой проблемы необходимо организовать динамическую маршрутизацию, учитывающую доступность отдельных частей системы. Каждый Sidecar Proxy и Service Discovery при подключении или отключении нового приложения проверяет его доступность, пытаясь сделать запрос по указанному в базе данных адресу. Полученный результат записывается в специальную таблицу Routes и используется в процессе маршрутизации. Данная таблица представляет ориентированный граф, описанный в виде матрицы смежности.

Основная задача Sidecar Proxy – перенаправление трафика от внешнего приложения. В зависимости от вида этого приложения должны быть применены разные правила маршрутизации. Для управления данным поведением используется сущность тип агента.

В условиях работы Service Mesh модуля внутри системы мониторинга, работающей в корпоративной распределенной системе, необходимо учитывать, что каждый отдельный запрос может перенаправляться в разные зоны, к разным адресатам.

Для настройки правил маршрутизации в зависимости от состояния флага CurrentArea адресат подходящего типа в необходимой зоне определяется, используя матрицу смежности, сохраненную в таблице routes. Кратчайший путь до адресата определяется с использованием алгоритма Дейкстры. Полученный путь сохраняется в HTTP-заголовок. По HTTP-заголовку proxy = true определяют, что запрос поступил не от сервиса и его нужно перенаправить по указанному пути. В случае невозможности отправки данных по указанному маршруту запрос сохраняется в память. При возобновлении связи с адресатом запрос будет отправлен заново, используя механизм асинхронной гарантированной доставки.

Маршрутизация внутри системы невозможна при отсутствии информации о текущем состоянии элементов системы. Для обеспечения своевременного обновления данных в системе применяется соединение в реальном времени по протоколу web-socket. В качестве инструмента для создания web-socket концентраторов был применен инструмент SignalR.

Существует три типа команд, вызывающих обновление данных внутри Sidecar Proxy:

1. Подключение нового агента. При подключении нового агента информация о подключении передается клиентам SignalR, каждый Sidecar Proxy в данной зоне проверяет доступность подключенного узла и обновляет матрицу смежности.

2. Обновление матрицы смежности. При обновлении матрицы смежности каждый Sidecar Proxy получает команду на загрузку актуальных данных.

3. Отключение агента. При отключении Sidecar Proxy от сети каждый Sidecar перепроверяет его доступность и обновляет матрицу смежности. В данном случае отключение от Service Discovery не означает, что элемент системы недоступен для других участников сети.

Результаты исследования и их обсуждение

Ниже приведен пример работы системы со следующей топологией (рис. 3).

Сеть за Service Discovery 1 относится к зоне 1, сеть за Service Discovery 2 к зоне 2. Сети ограничены межсетевым экраном, доступ наружу есть только у Service Discovery серверов. Сервис типа C – это подвижный элемент системы, способный перемещаться из одной зоны в другую. Он является клиентом сервисов A и B. Настройка маршрутизации изображена в таблице.

От внешнего приложения, в данном случае от консольной утилиты curl, совершаются запросы по указанным маршрутам через сервис типа C. На рис. 4 изображены пути прохождения запросов в случае, если агент находится в зоне 1 и в зоне 2.

При перемещении сервиса C из зоны 1 в зону 2 поведение системы меняется

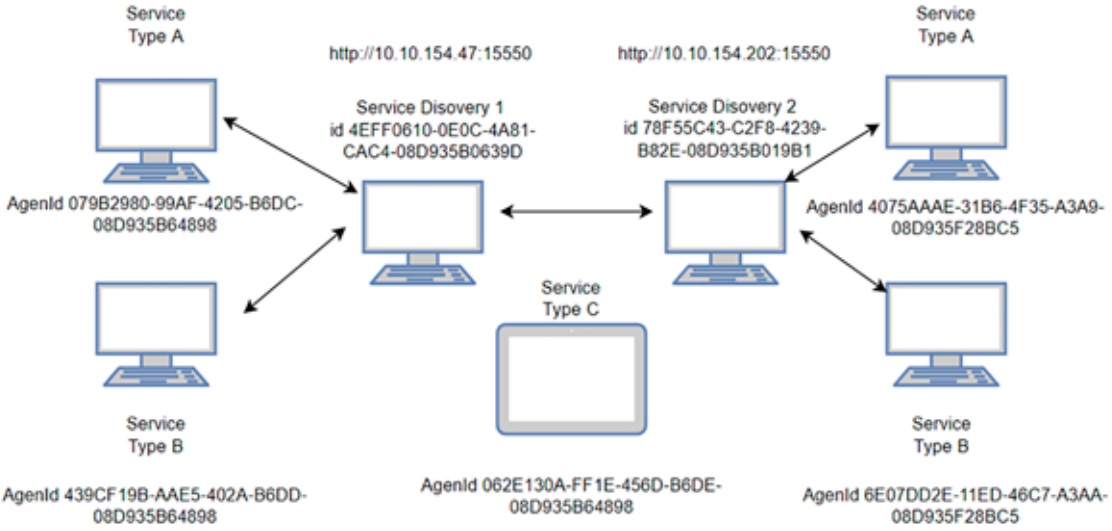

Рис. 3. Пример топологии Service Mesh сети

Настройка маршрутизации

agentType	destinationType	route	method	currentArea	direct
Service Type C	Service Type B	api/test	GET	1	1
	Service Type A	api/2test	GET	1	0
	Service Type B	api/test	POST	0	0

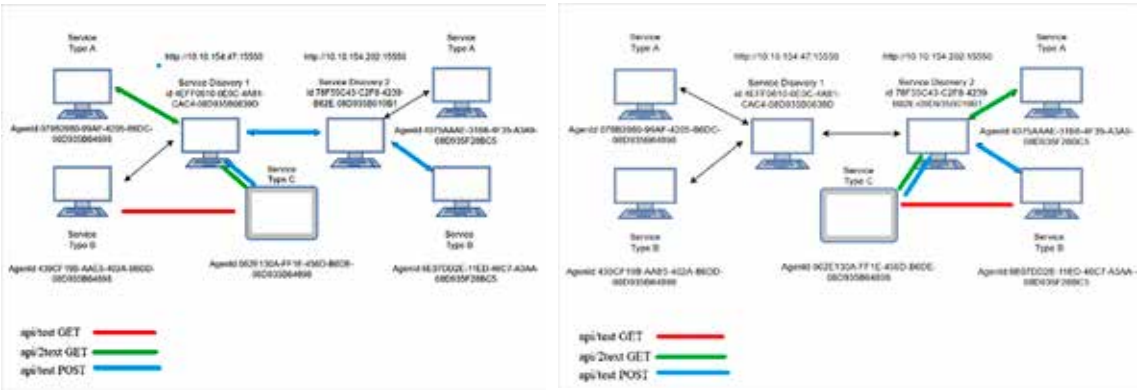

Рис. 4. Пути прохождения запросов для зоны 1 (слева) и зоны 2 (справа)

Заключение

Спроектированный Service Mesh позволит облегчить работу в условиях территориальной распределенности за счет организации маршрутизации, упростить процессы развертывания за счет механизма регистрации в Service Mesh сети и повысить стабильность за счет организации работы в условиях отсутствия связи с центральным сервером.

Разработанный Service Mesh модуль может быть применен отдельно от системы мониторинга как связующее звено для организации сетевого взаимодействия любой распределенной системы. Практическая ценность использования результатов работы заключается в повышении производительности труда работников служб поддержки, использующих систему мониторинга; минимизировании расходов бизнеса; обеспечении непрерывности бизнес-процессов; своевременном реагировании на инциденты за счет перенаправления информации об инцидентах в ближайший филиал компании.

Список литературы

1. Flo A. Does the Service Mesh spell the end for Middleware? [Электронный ресурс]. URL: https://www.cloudops.com/blog/does-the-service-mesh-spell-the-end-for-middleware (дата обращения: 06.20.2021).

2. Завьялова Н.Б. Корпоративные информационные системы управления. М.: Инфра-М, 2015. 464 с.

3. Бернс Б. Распределенные системы. Паттерны проектирования. М.: Питер, 2019. 512 с.

4. Maarten van Steen, Andrew S. Tanenbaum. A brief introduction to distributed systems // Computing 98. 2016. P. 967–1009.

5. Джулиан М. Мониторинг на практике. М.: OReilly, 2014. 170 с.