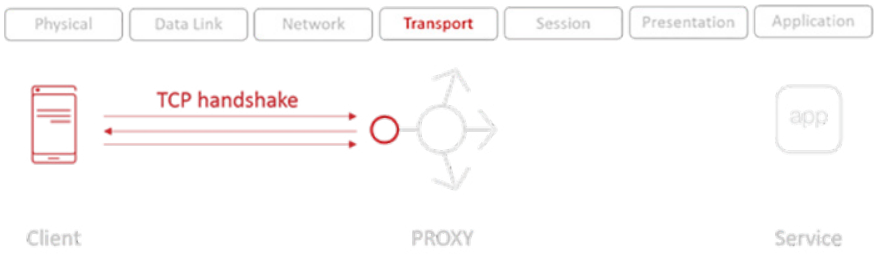
ПРАКТИЧЕСКОЕ РУКОВОДСТВО ПО ПРОТОКОЛУ: КАК ПРОКСИ-СЕРВЕР ВЗАИМОДЕЙСТВУЕТ С HTTP

Лори МакВитти (Lori MacVittie), главный технический евангелист F5 Networks, США

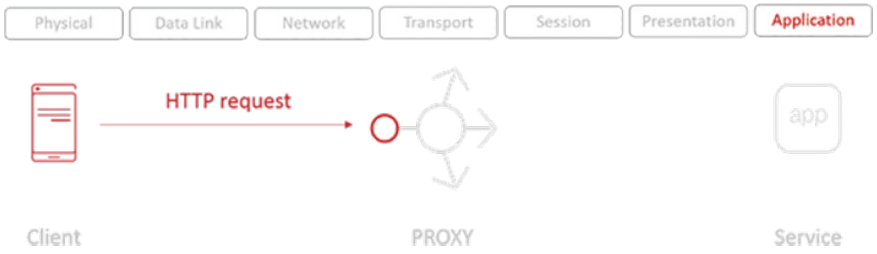
Внедрение контейнеров и кластеризации (с ее замкнутой экосистемой балансировщиков, средств контроля входа и прокси-серверов) может оказаться довольно сложной задачей. Это связано с тем, что они встраиваются в протокол TCP, ориентированный на поддержание соединения. Поверх TCP для обмена данными используется протокол HTTP, ориентируемый на обмен сообщениями. Чтобы понять, каким образом использовать прокси-серверы в новых системах, нужно хорошо разобраться в основе принципа их работы «запрос-ответ» в случае использования программного прокси-сервера.

Прокси-серверы, которые достаточно «умны» для работы на уровне 7 (HTTP или «уровень приложений»), позволяют использовать множество способов маршрутизации данных – от выбора ресурса для обслуживания запроса (балансирования и маршрутизации нагрузки) до модификации HTTP-заголовков. Для этого прокси-серверу приходится перехватывать сообщения и проверять их. Это означает, что он работает и на уровне 4 (TCP), и на уровне 7 (HTTP). Кроме того, это значит, что прокси-серверы – посредники. Они являются «промежуточным уровнем сети», представляя собой удобное место для контроля над принятием решений по поводу обработки запросов (и, соответственно, отправки ответов).

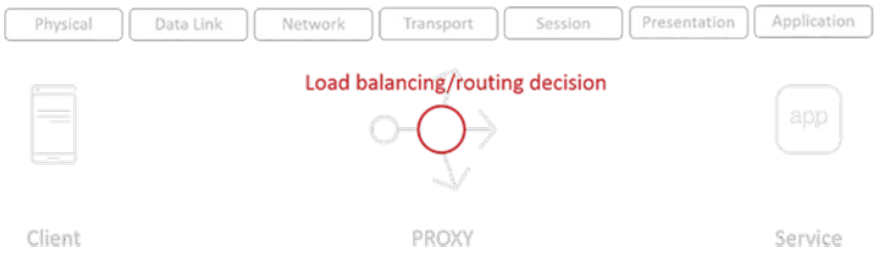
Ниже показан порядок обработки запроса HTTP 1.x (незащищенного), поступающего к службе и обратно от нее через прокси-сервер (HTTP/2 меняет все, в будущем ему придется посвятить отдельный блог).



Шаг 1. Пользователю (Приложение) необходимо обратиться к службе (Сервер приложения). DNS передает клиенту IP-адрес, а клиент использует его для начала сеанса TCP. На самом деле этот сеанс инициируется прокси-сервером. Здесь он просто устанавливает соединение, ничего больше не делая. На данном этапе можно применить базовые средства защиты IP, например, черные списки для предотвращения подключения известных зловредов или предоставления доступа только разрешенным сетям. На уровне прокси-сервера может быть обеспечена более совершенная защита – некоторые из них способны выявлять вредоносную деятельность на основании поведения TCP.



Шаг 2. Пользователь, установив подключение по TCP, отправляет HTTP-запрос. Это может быть запрос на подключение к API или веб-странице. С точки зрения HTTP – и первый, и второй являются HTTP-запросами. Сначала поступают заголовки HTTP, а за ними и необходимая информация.

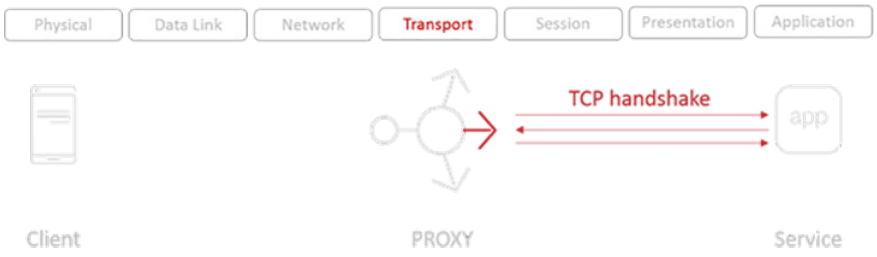


Шаг 3. Именно на этом шаге прокси-сервер начинает работать на полную. Он может выполнять самые различные операции, начиная с выбора службы или ресурса для выдачи ответа на запрос. Это делается путем применения какого-либо алгоритма балансировки нагрузки (Round Robin, минимального количества подключений и т.п.) или путем выбора ресурса на основе некой таблицы «маршрутов» 7-го уровня.

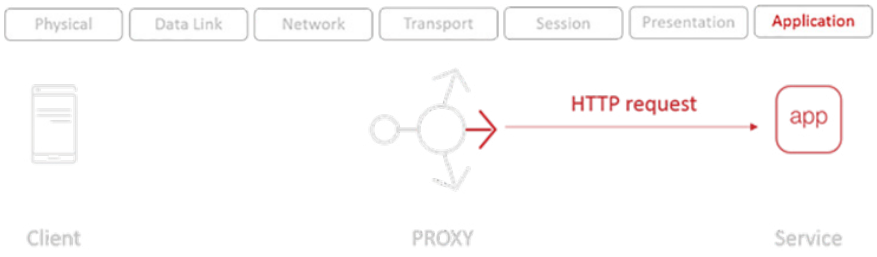
Например, сопоставление значения параметров типа «версии» с заголовками HTTP, а также использование их для определения службы в кластере контейнера, которая должна получить этот запрос. Виртуальный хостинг с маршрутизацией по имени работает аналогичным образом, используя заголовок хоста HTTP и сопоставляя его с конкретным сервером.

В этот момент появляется возможность выполнения различных проверок безопасности сообщения HTTP (с полезной информацией), например, можно просканировать его на наличие зловредного содержимого, указывающего, например, на атаку SQLi или XSS.

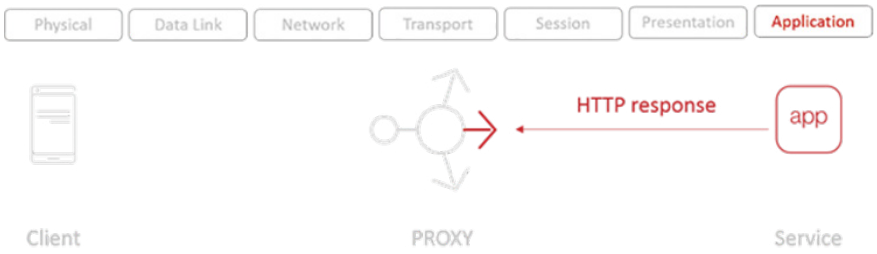
Кроме того, появляется возможность подставить заголовки HTTP. Широко распространена практика добавления заголовка X-Forwarded-For (XFF) для того, чтобы оставить фактический IP-адрес клиента для использования приложением.



Шаг 4. После выбора ресурса или службы прокси-сервер должен установить соединение со службой по TCP (сервер приложения). Именно это разделение «клиентской» и «серверной» сторон является главным фактором, обеспечивающим возможность проведения более детальных проверок безопасности и бизнес-логики на прокси-сервере. Кроме того, это означает наличие двух абсолютно разных сетевых стеков, каждый из которых можно настраивать и оптимизировать независимо. Это радикальным образом повышает производительность, так как клиенты и службы/приложения часто имеют разные сетевые профили/настройки.

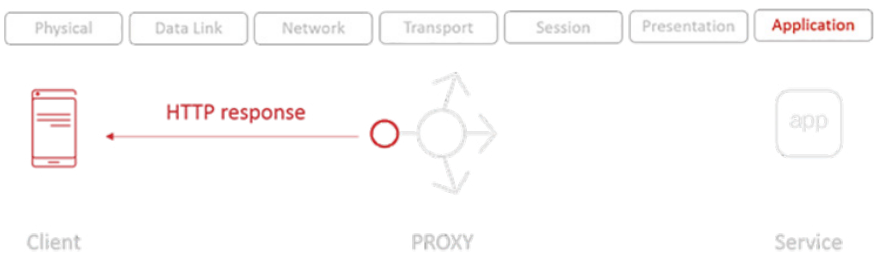


Шаг 5. Первоначальный HTTP-запрос (с учетом всех изменений, внесенных прокси-сервером) передается службе/ресурсу.



Шаг 6. Прокси-сервер получает ответ. Как мы видели в шаге 3, получив ответ прокси-сервер может проверить и оценить его. Именно на этом шаге обычно выполняются задачи по обеспечению безопасности – например, предотвращение утечки данных. Ответы могут оцениваться также при проверке кодов состояний HTTP, что позволяет осуществлять дополнительные действия, вроде повторной попытки пересылки неудачного запроса другой службе.

Кроме того, прокси-сервер может собирать телеметрическую информацию о производительности. Например, при получении запроса производится пассивный мониторинг, позволяющий прокси-серверу отслеживать время и состояние отклика по пулу своих ресурсов (серверов). Эти данные можно использовать не только для создания панелей статистики, мониторинга и составления отчетности о производительности, но и для использования в качестве входного параметра алгоритмов балансирования нагрузки, решения которых основываются на времени отклика.



Шаг 7. Теперь, наконец, обеспечивается выполнение первоначального запроса и прокси-сервер возвращает клиенту ответ, полученный от сервера приложения. TCP-соединение между клиентом и прокси-сервером (как правило) остается открытым для обработки дальнейших запросов. Со временем соединения, в зависимости от настроек, разрываются по превышению лимита времени ожидания. Это одно из тех значений, которые можно настраивать для повышения производительности, исходя из закономерностей использования приложения, доступ к которому организован через прокси-сервер.

Вот и все! Основной поток HTTP 1.x с промежуточным прокси-сервером выполняет балансирование нагрузки и/или функции безопасности. Понимание основного потока HTTP 1.x поможет разобраться в том, где лучше всего применять политики и развертывать дополнительные службы приложений – например, для идентификации и защиты приложений.

F5 Networks, Inc.
401 Elliott Avenue West, Seattle, WA 98119
888-882-4447 f5.com

Americas
info@f5.com

Asia-Pacific
apacinfo@f5.com

Europe/Middle-East/Africa
emeainfo@f5.com

Japan
f5j-info@f5.com