Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

Ю.С. Белов, Е.А. Черепков

ИСПОЛЬЗОВАНИЕ СИСТЕМЫ В КАЧЕСТВЕ ПРОКСИ-СЕРВЕРА

Методические указания к лабораторной работе

по дисциплине «Операционные системы»

Калуга – 2018

УДК 004.62
ББК 32.972.1
Б435

Методические указания составлены в соответствии с учебным планом КФ МГТУ им. Н.Э. Баумана по направлению подготовки 09.03.04 «Программная инженерия» кафедры «Программного обеспечения ЭВМ, информационных технологий».

Методические указания рассмотрены и одобрены:

- Кафедрой «Программного обеспечения ЭВМ, информационных технологий» (ИУ4-КФ) протокол № 51.4/6 от «20» февраля 2019 г.

Зав. кафедрой ИУ4-КФ _____к.т.н., доцент Ю.Е. Гагарин

- Методической комиссией факультета ИУ-КФ протокол № 9 от «04» 03 2019 г.

Председатель методической
комиссии факультета ИУ-КФ _____к.т.н., доцент М.Ю. Адкин

- Методической комиссией
КФ МГТУ им.Н.Э. Баумана протокол №5 от «5» 03 2019 г.

Председатель методической комиссии
КФ МГТУ им.Н.Э. Баумана _____д.э.н., профессор О.Л. Перерва

Рецензент:
к.т.н., доцент кафедры ИУ3-КФ _____ А.В. Финошин

Авторы
к.ф.-м.н., доцент кафедры ИУ4-КФ _____ Ю.С. Белов
ассистент кафедры ИУ4-КФ _____ Е.А. Черепков

Аннотация

Методические указания к выполнению лабораторной работы по курсу «Операционные системы» содержат общие сведения о назначении и применении прокси-сервера, описан принцип работы прокси-сервера а также его установка и настройка в ОС FreeBSD.

Предназначены для студентов 3-го курса бакалавриата КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Настоящие методические указания составлены в соответствии с программой проведения лабораторных работ по курсу «Операционные системы» на кафедре «Программное обеспечение ЭВМ, информационные технологии» факультета «Информатика и управление» Калужского филиала МГТУ им. Н.Э. Баумана.

Методические указания, ориентированные на студентов 3-го курса направления подготовки 09.03.04 «Программная инженерия», содержат краткое описание работы прокси-сервера, основные принципы и руководство по его настройке.

Методические указания составлены для ознакомления студентов с операционной системой FreeBSD и овладения начальными навыками по настройке и работе с прокси-сервером. Для выполнения лабораторной работы студенту необходимы минимальные знания по установке операционных систем.

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ

Целью выполнения лабораторной работы является получение практических навыки по настройке прокси-сервера Squid под ОС FreeBSD.

Основными задачами выполнения лабораторной работы являются:

1. Научиться получать и устанавливать прокси-сервер Squid под ОС FreeBSD
2. Научиться настраивать и управлять прокси-сервером Squid под ОС FreeBSD

Результатами работы являются:

1. Настроенный прокси-сервер Squid в ОС FreeBSD.
2. Подготовленный отчет.

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

Прокси-сервер (от англ. *proxy* — «представитель, уполномоченный») — служба в компьютерных сетях, позволяющая клиентам выполнять косвенные запросы к другим сетевым службам. Сначала клиент подключается к прокси-серверу и запрашивает какой-либо ресурс (например, файл), расположенный на другом сервере. Затем прокси-сервер либо подключается к указанному серверу и получает ресурс у него, либо возвращает ресурс из собственного кэша (в случаях, если прокси имеет свой кеш). В некоторых случаях запрос клиента или ответ сервера может быть изменён прокси-сервером в определённых целях.

Из многочисленных значений английского слова *proxy* в данном контексте применимы такие: «доверенное лицо», «полномочный представитель». То есть некто, кто действует от вашего имени по вашему поручению вместо вас. В компьютерах прокси – это программа, которая передает запросы ваших программ (браузеров и других) в интернет, получает ответы и передает их обратно. Необходимость в такой программе возникает обычно, если с пользовательского компьютера невозможно работать в интернете непосредственно напрямую из-за того, что у него нет прямого подключения к интернету (модема, например), но есть на другом компьютере в его сети. Тогда на этом другом компьютере ставят программу прокси, а все остальные компьютеры в локальной сети настраивают таким образом, чтобы работа велась через прокси. Сейчас через прокси умеют работать практически все популярные интернет-программы. Это значит что все пользователи локальной сети могут получить полноценный доступ в интернет, если хотя бы один из них этот доступ уже имеет.

Чаще всего прокси-серверы применяются для следующих целей:

- Обеспечение доступа с компьютеров локальной сети в Интернет.
- Кэширование данных: если часто происходят обращения к одним и тем же внешним ресурсам, то можно держать их

копию на прокси-сервере и выдавать по запросу, снижая тем самым нагрузку на канал во внешнюю сеть и ускоряя получение клиентом запрошенной информации.

- Сжатие данных: прокси-сервер загружает информацию из Интернета и передаёт информацию конечному пользователю в сжатом виде. Такие прокси-серверы используются в основном с целью экономии внешнего трафика.
- Защита локальной сети от внешнего доступа: например, можно настроить прокси- сервер так, что локальные компьютеры будут обращаться к внешним ресурсам только через него, а внешние компьютеры не смогут обращаться к локальным вообще (они «видят» только прокси-сервер). См. также *NAT*.
- Ограничение доступа из локальной сети к внешней: например, можно запретить доступ к определённым веб-сайтам, ограничить использование интернета каким-то локальным пользователям, устанавливать квоты на трафик или полосу пропускания, фильтровать рекламу и вирусы.
- Анонимность доступа к различным ресурсам. Прокси-сервер может скрывать сведения об источнике запроса или пользователе. В таком случае целевой сервер видит лишь информацию о прокси-сервере, например, IP-адрес, но не имеет возможности определить истинный источник запроса. Существуют также *искажающие прокси-серверы*, которые передают целевому серверу ложную информацию об истинном пользователе.

Работа компьютеров локальной сети в Интернете без прокси

Такая работа требует выполнения определенных дополнительных условий и имеет свои минусы в сравнении с работой через прокси. Какие условия: каждому компьютеру должен быть выдан персональный IP-адрес в сети Интернет, и построена схема маршрутизации так, что пакеты будут попадать именно на этот компьютер. Это невозможно сделать без участия провайдера. Провайдеры идут на это, но, как правило, за

отдельную плату. Например, $5 за каждый IP в месяц. Это хлопотно, дорого и снижает уровень безопасности в целом: каждый компьютер вашей сети станет потенциальной мишенью хакеров, вирусных атак и прочих «прелестей» Интернета. При правильной настройке компьютеров это не очень страшно, но рядовые пользователи не склонны следить за безопасностью своих компьютеров, значит, будут дополнительные хлопоты у администраторов сети. Возможностей контролировать работу пользователей у администратора будет немного, так как система децентрализована. Кстати, при таком способе подключения тоже нужна программа-посредник на том компьютере, который непосредственно подключен к интернету. Но при наличии реальных IP-адресов эта программа – обычный роутер (маршрутизатор) IP-пакетов, он является частью операционной системы. И к этой программе название «прокси» не применяют. Важное отличие маршрутизатора от прокси – при использовании маршрутизатора IP-пакеты остаются без изменений, в них сохраняются исходные адреса компьютеров ЛС. А прокси всегда работает от своего адреса, а адреса клиентов не передаются, т.к. недоступны из интернета. Маршрутизатор, меняющий адреса, уже является прокси (его называют *NAT-proxy*, NAT (Network Address Translation).

ВИДЫ ПРОКСИ

Упомянутый выше NAT-proxy – самый простой вид прокси. Теперь он даже входит в состав Windows 2000 и Windows XP. Там он называется «Общий доступ к подключению интернета» и включается галочкой в свойствах модемного соединения. Этот прокси работает прозрачно для пользователя, никаких специальных настроек в программах не требуется. Но на этом удобства этого прокси заканчиваются. Влиять на работу «общего доступа» Windows (например, ограничивать список доступных сайтов для отдельных пользователей) вы не сможете. Другие NAT-proxy могут быть более гибкими, но их общая проблема – универсальность. Они не «вникают» в тонкости тех прикладных протоколов, которые через себя пропускают, поэтому и не имеют средств управления ими.

Специализированные прокси (для каждого протокола интернета свой вид прокси) имеют ряд преимуществ и с точки зрения администраторов, и с точки зрения пользователей. Ниже перечислены виды специализированных прокси.

HTTP-прокси

HTTP-прокси – самый распространенный. Он предназначен для организации работы браузеров и других программ, использующих протокол HTTP. Браузер передает прокси- серверу URL ресурса, прокси-сервер получает его с запрашиваемого веб-сервера (или с другого прокси-сервера) и отдает браузеру. У HTTP-прокси широкие возможности при выполнении запросов:

Можно сохранять полученные файлы на диске сервера. Впоследствии, если запрашиваемый файл уже скачивался, то можно выдать его с диска без обращения в интернет – увеличивается скорость и экономится внешний трафик (который может быть платным). Эта опция называется кэшированием – именно её очень любят администраторы и пользователи – настолько, что считают её главной функцией прокси. Однако приводимые оценки экономии (в описаниях встречалось от 30 до 60%) слишком оптимистичны, не верьте им. На деле получается не более 10-15% – современный

интернет очень динамичен, страницы часто меняются, зависят от работающего с ними пользователя и т.д. – такие данные кэшировать нельзя, веб-серверы обычно вставляют в HTTP-заголовки специальные указания об этом, чтобы браузеры и прокси имели это в виду. Хотя многие прокси-серверы можно настроить так, чтобы эти указания частично игнорировались – например, перечитывать страницу не чаще одного раза в день.

- Можно ограничивать доступ к ресурсам. Например, завести «черный список» сайтов, на которые прокси не будет пускать пользователей (или определенную часть пользователей, или в определенное время и т.д.). Ограничения можно реализовать по-разному. Можно просто не выдавать ресурс – например, выдавая вместо него страницу «запрещено администратором» или «не найдено». Можно спрашивать пароль и авторизованных пользователей допускать к просмотру. Можно, не спрашивая пароля, принимать решение на основании адреса или имени компьютера пользователя.
- Можно выдавать не тот ресурс, который запрашивается браузером. Например, вместо рекламных баннеров и счетчиков показывать пользователям прозрачные картинки, не нарушающие дизайн сайта, но существенно экономящие время и трафик за счет исключения загрузки картинок извне.
- Можно ограничивать скорость работы для отдельных пользователей, групп или ресурсов. Например, установить правило, чтобы файлы *.mp3 качались на скорости не более 1кб/сек, чтобы предотвратить забивание вашего интернет- канала трафиком меломанов, но не лишать их полностью этого удовольствия. Эта возможность, к сожалению, есть не во всех прокси.
- Ведутся журналы работы прокси – можно подсчитывать трафик за заданный период, по заданному пользователю, выяснять популярность тех или иных ресурсов и т.д.
- Можно маршрутизировать веб-запросы – например, часть направлять напрямую, часть через другие прокси (прокси

провайдера, спутниковые прокси и т.д.). Это тоже помогает эффективнее управлять стоимостью трафика и скоростью работы прокси в целом.

FTP-прокси

FTP-прокси бывает двух основных видов в зависимости от протокола работы самого прокси. С ftp-серверами этот прокси, конечно, всегда работает по протоколу FTP. А вот с клиентскими программами – браузерами и ftp-клиентами *(CuteFTP, FAR*, и др.) прокси может работать как по FTP, так и по HTTP. Второй способ удобнее для браузеров, т.к. исторически является для них «родным». Браузер запрашивает ресурс у прокси, указывая протокол целевого сервера в *URL – http* или *ftp*. В зависимости от этого прокси выбирает протокол работы с целевым сервером, а протокол работы с браузером не меняется – HTTP. Поэтому, как правило, функцию работы с FTP-серверами также вставляют в HTTP- прокси, т.е. HTTP-прокси, описанный выше, обычно с одинаковым успехом работает как с HTTP, так и с FTP-серверами. Но при «конвертации» протоколов FTP<->HTTP теряется часть полезных функций протокола FTP. Поэтому специализированные ftp- клиенты предпочитают и специальный прокси, работающий с обеими сторонами по FTP. В *Eserv* и *Eproxy* мы называем этот прокси FTP-gate, чтобы подчеркнуть отличие от FTP- прокси внутри HTTP-прокси. Также этот прокси называется в некоторых ftp-клиентах. Хотя встречаются и вносящие путаницу названия. Например, в программе *CuteFTP FTP- gate* называют *firewall*, хотя FireWall в общем случае – это вообще не прокси, а фактически программа обратного назначения – не для подключения к интернету, а для изоляции от него ;) Для прокси в FireWall оставляют специальные «дыры». FTP-gate поддерживают различные способы указания в FTP-протоколе целевого сервера, с которым FTP-клиент хочет работать, в настройке FTP-клиентов обычно предлагается выбор этого способа, например, *USER user@site, OPEN site*, и т.д. – способ указания сервера, с которым производится работа. Такое многообразие связано с тем, что

нет общепринятого стандарта на этот вид прокси, и применяются такие хитрые добавки к стандартным командам FTP-протокола.

HTTPS-прокси

HTTPS-прокси – фактически часть HTTP-прокси. S в названии означает «*secure*», т.е. безопасный. Не смотря на то, что программно эту часть HTTP-прокси, обычно HTTPS выделяют в отдельную категорию (и есть отдельное поле для него в настройке браузеров). Обычно этот протокол – безопасный HTTP – применяют, когда требуется передача секретной информации, например, номеров кредитных карт. При использовании обычного HTTP-прокси всю передаваемую информацию можно перехватить средствами самого прокси (т.е. это под силу администратору ЛС) или на более низком уровне, например, *tcpdump* (т.е. и администратор провайдера и любого промежуточного узла и вообще любой человек, имеющий физический доступ к маршрутам передачи ваших данных по сети, может при большом желании узнать ваши секреты). Поэтому в таких случаях применяют *secure* HTTP – всё передаваемое при этом шифруется. Прокси-серверу при этом дается только команда «соединится с таким-то сервером», и после соединения прокси передает в обе стороны шифрованный трафик, не имея возможности узнать подробности (соответственно и многие средства управления доступом – такие как фильтрация картинок – не могут быть реализованы для HTTPS, т.к. прокси в этом случае неизвестно, что именно передается). Собственно в процессе шифрации/дешифрации прокси тоже участия не принимает – это делают клиентская программа и целевой сервер. Наличие команды «соединиться с таким-то сервером» в HTTPS-прокси приводит к интересному и полезному побочному эффекту, которым все чаще пользуются разработчики клиентских программ. Так как после соединения с указанным сервером HTTPS-прокси лишь пассивно передает данные в обе стороны, не производя никакой обработки этого потока вплоть до отключения клиента или сервера, это позволяет использовать прокси для передачи почти любого TCP-протокола, а не только HTTP. То есть HTTPS-прокси одновременно

является и простым POP3-прокси, SMTP-прокси, IMAP-прокси, NNTP-прокси и т.д. – при условии, что соответствующая клиентская программа умеет так эксплуатировать HTTPS-прокси (увы, далеко не все еще это умеют, но есть вспомогательные программы, «заворачивающие» трафик обычных клиентов через HTTPS-прокси). Никаких модификаций целевого сервера не требуется. Фактически HTTPS-прокси является программируемым *mapping-proxy*, как и *Socks-proxy*.

Mapping-прокси

Mapping-прокси – способ заставить работать через прокси те программы, которые умеют работать с интернетом только напрямую. При настройке такого прокси администратор создает как бы «копию» целевого сервера, но доступную через один из портов прокси- сервера для всех клиентов локальной сети – устанавливает локальное «отображение» заданного сервера. Например, пользователи локальной сети хотят работать с почтовым сервером *mail.ru* не через браузер, а с использованием почтовой программы *Outlook Express* или *TheBat*. Эти программы не умеют работать через прокси (кроме случая, когда *Outlook* получает почту по HTTP с *hotmail.com* – тогда он, как и браузер, пользуется HTTP-прокси). Простейший способ работать с *mail.ru* по POP3 через прокси – установить локальное отображение сервера *pop.mail.ru*. И в *Outlook*'ах вместо *pop.mail.ru* написать имя прокси-сервера и порт отображения. *Outlook* будет соединяться с прокси-сервером («думая», что это почтовый сервер), а прокси при этом будет соединяться с *pop.mail.ru* и прозрачно передавать всю информацию между *Outlook* и *pop.mail.ru*, таким образом

«превращаясь» на время соединения в POP3-сервер. Неудобство *mapping*-прокси в том, что для каждого необходимого внешнего сервера нужно вручную устанавливать отдельный порт на прокси. Но зато не требуется модификация ни серверов, ни клиентов. Особенно это помогает в случае необходимости «проксирования» многочисленных

«доморощенных» протоколов, реализованных в играх или финансовых программах. Почему-то они часто игнорируют существование прокси и стандартных протоколов. Такие программы можно «обмануть» и направить через прокси практически всегда, если они не делают другой глупости – передачи клиентского IP-адреса внутри протокола и пытаются с ним соединяться напрямую еще раз (что невозможно, т.к. локальные адреса недоступны извне).

Socks-прокси

Socks-прокси. SOCKS5 – протокол для прокси-сервера, позволяющий пропускать через прокси почти любой прикладной TCP- или UDP-протокол. SOCKS4 – старая версия протокола, имеет ряд ограничений – в частности, не поддерживается передача имени хоста вместо IP-адреса.

Кэш

В интернете неизбежны перегрузки и «заторы», что объясняется наличием следующих проблем: низкие скорости соединения; непредсказуемые технические характеристики; ограничения по полосе частот; Web-сайты, перегруженные заказами. Один из способов решения этих проблем связан с использованием устройств кэширования (от слова *cache* - тайник, склад, запас).

Средства кэширования включают в состав Web-браузеров, что позволяет запоминать некоторые Web-страницы, к которым обращался пользователь компьютера, для их последующего повторного использования. Точно такой же принцип заложен и в любое устройство Web-кэширования: Web-контент перемещается в некий сетевой кэш поближе к пользователям, нуждающимся в нем, вследствие чего уменьшается число участков маршрутизации или коммутации, через которые он должен пройти. В случае корпоративных пользователей наиболее близкое расположение такого контента - в самой корпоративной сети. Сетевые кэши работают на тех же принципах, что и браузеры, однако они выбирают контент, анализируя активность сотен и тысяч пользователей, а не одного, как в случае с браузером.

RFC 2187 описывает протокол ICP (Internet Cache Protocol), который позволяет осуществлять иерархическое соединение кэшей. Он определяет порядок обмена информацией между кэшами, находящимися в состоянии подчинения.

ICP прежде всего используется в иерархии кэшей для поиска определенных объектов в братских кэшах. Если *squid* не находит нужного документа, то посылает ICP запрос братским кэшам, которые в свою очередь отвечают ICP ответами «HIT» («попадание») или «MISS» («промах»). Затем кэш использует ответы для выбора, при помощи какого кэша разрешать свои ответы MISS.

Прокси и кэш

Кэш-серверы и прокси-серверы - не одно и то же. Кэширование по-прежнему остается одной из функций прокси-серверов. Однако повышение спроса на специализированное кэширование приводит к тому, что кэш-серверы все чаще выпускаются в качестве отдельных продуктов. Так, продукт *CacheQube* компании *Cobalt Networks* представляет собой устройство, которое просто устанавливается между локальной сетью и маршрутизатором для осуществления прозрачного кэширования. *Streaming Media Cache* фирмы *Inktomi* и *MediaMail* производства *InfoLibria* представляют собой кэши, специально предназначенные для обработки потоковых аудио и видео.

Прокси-кэш

Кэш-серверы изучают активность, перехватывая запросы одним из двух способов: путем прозрачного кэширования или прокси-кэширования. Прозрачный кэш-сервер

«просеивает» через себя весь проходящий трафик и поэтому не требует модификации установок конечного клиента. Он устанавливается обычно перед маршрутизатором, соединенным с интернетом.

В случае прокси-кэша сетевые администраторы конфигурируют пользовательские браузеры так, чтобы они направляли запросы на контент непосредственно в кэш. Затем прокси-кэш-сервер

запрашивает нужный контент от имени пользователя. Это позволяет сетевым администраторам также ограничить число сайтов, на которые могут заходить пользователи. Такой подход более сложен, поскольку требует конфигурирования каждого клиента. Кроме того, если в этом случае прокси-кэш-сервер выйдет из строя, пользователи не смогут обращаться к Web.

Таким образом, прокси-кэш - некое средство в прокси-сервере, которое кэширует поступающие Web-страницы на жестком диске. Если страница, запрашиваемая браузером, уже находится в прокси-кэше, то она отыскивается в нем, а не в интернете. Так случилось, что прокси-кэш-серверами называют практически все устройства кэширования, независимо от их расположения относительно потока информации, а «прозрачное» кэширование стало лишь одним из режимов работы прокси-кэш-сервера.

Существует несколько подходов к реализации архитектуры прокси-кэша, которые принято называть моделями. Выбор той или иной модели определяется размещением прокси-кэша, его главным назначением и природой трафика. Помимо прозрачного кэширования, существуют еще следующие архитектуры (модели) прокси-кэш-сервера:

- Прямой прокси-кэш. При такой конфигурации запросы пользователей на своем пути к Web-серверу проходят через кэш. Если кэш содержит запрашиваемый документ, этот документ отправляется пользователю. В противном случае сервер работает как прокси, извлекая нужный контент из Web-сервера.
- Обратный прокси-кэш, или «серверный ускоритель». Кэш может быть также сконфигурирован как быстрый Web-сервер для ускорения более медленных традиционных Web-серверов. При этом документы, хранящиеся в кэше, обрабатываются с высокой скоростью, в то время как документы, не занесенные в кэш (обычно динамический контент) запрашиваются при необходимость из исходных Web-серверов. Такая кэширующая система располагается перед одним или

несколькими Web-серверами, перехватывая запросы и действуя наподобие прокси. Эти прокси-кэш-серверы могут размещаться по всей сети, формируя некую распределенную сеть сайтов для хостирования контента. Дополнительное достоинство данной схемы связано с возможностью балансировки нагрузки и динамического зеркалирования.

Продукты

В настоящее время все прокси-кэш-серверы, представленные на рынке, могут быть классифицированы следующим образом:

- Прокси-кэш-серверы со специализированной ОС (*CacheFlow*, *Network Alliance* и *Cisco Systems*). В таких серверах само ядро ОС разработано с учетом требований, предъявляемых к прокси-кэш-серверам реальными условиями их использования - продолжительные потоки данных, файлы большого размера и длительные сеансы связи.
- Изделия, представляющие собой специализированные программно-аппаратные средства со стандартной ОС (*Cobalt Networks*).
- Стандартные серверы, несколько оптимизированные для использования в качестве Web-серверов, на которых установлено ПО кэширования. К этой категории относятся прокси-кэш-серверы компании *Compaq* с ПО *Novell Internet Caching System* и корпорации *Intel* с ПО *Inktomi Traffic Server Engine*. Такие серверы используются обычно в локальных сетях средних размеров. По итогам испытаний, проведенных *IRCache* (см. «Сетевой журнал», № 1/2000), лучшим был признан продукт именно этого класса, который состоял из аппаратного комплекса *Dell* и ПО *Novell*.

CacheFlow специализируется непосредственно на системах кэширования, для чего ею была разработана собственная ОС *CachOS*, оптимизирующая функции кэширования. В продуктах этой фирмы реализована технология *Object Pipelining*, которая позволяет организовать быстрый доступ к контенту с первого раза, ликвидируя

значительную часть задержек на пути от Web-браузера клиента до удаленного Web-сервера провайдера. Возникновение таких задержек связано с тем, что Web-страницы, как правило, состоят из множества объектов и для каждого такого объекта обычно должна сначала открываться TCP-сессия, после чего уже следует получение HTTP-запроса. Вместо последовательного получения объектов *Object Pipelining* сразу открывает такое количество TCP-сессий, какое удаленный сервер может позволить, и получает объекты параллельно. После этого объекты доставляются от устройства прямо пользователю настолько быстро, насколько браузер пользователя может их запрашивать.

Также реализован алгоритм адаптивного обновления (*Adaptive Refresh*), предназначенный для ускорения обработки повторных запросов. Так как содержание Web-серверов постоянно меняется, прокси-кэш-сервер должен содержать временное хранилище контента в актуальном состоянии. В традиционном решении для того, чтобы гарантированно доставить пользователю актуальные данные, прокси-кэш-сервер должен обязательно произвести проверку контента на исходном сервере. С другой стороны, чтобы доставить данные быстро, сервер не должен ждать, пока пользователь запросит контент, перед тем как сервер обновит страницы. Если обновление контента происходит только в тот момент, когда пользователь посылает запрос, последний сталкивается со значительными задержками. Единственный метод доставки Web-страниц быстро и адекватно - это производить освежение контента асинхронно с запросами клиентов. Запатентованный алгоритм адаптивного обновления селективно обновляет содержимое кэша в зависимости от потребности. Для каждого объекта, хранящегося в кэше, формируются две модели - модель изменений и модель использования, из комбинаций которых формируются рациональные сценарии обновления. Эти сценарии динамически изменяются в зависимости от изменения модели.

Компания утверждает, что уменьшение используемой полосы достигает 60%, сокращение времени доставки контента пользователю (или провайдеру) или времени отклика сети доходит до десятикратного, разгрузка Web-серверов составляет до 70% запросов.

Еще одна особенность этих продуктов - хорошая совместимость с сетевым оборудованием Cisco, о которой говорят многие системные интеграторы в России.

Network Appliance - крупнейший поставщик NAS (Network Attached Storage), и, естественно, ее коньком являются технологии предоставления быстрого, надежного доступа по сети к большим объемам данных. Подходы, отработанные при проектировании устройств доступа, фирма перенесла и на свои прокси-кэш-серверы:

- отказоустойчивая высокопроизводительная архитектура (поддержка RAID, fibre channel, горячая замена блоков питания и вентиляторов);
- оптимизированная под RAID4 файловая система WAFL плюс кэширование запросов на запись в памяти типа NVRAM.

Продукты используют собственную ОС (*NetCache*) и поддерживают ICAP -- протокол, по которому прокси-кэш-сервер передает принятые данные на сторонний сервер, осуществляющий их мониторинг (например, дополнительную проверку на вирусы) и возвращающий обратно. Имеется поддержка *takeover*, когда два сервера работают в паре; если один вышел из строя, пользователи продолжают прозрачно работать через второй. Предусмотрена аутентификация пользователей через *Radius/LDAP/NTLM*, имеется поддержка *streaming media*.

Cisco Systems. В продукции применена ОС *Cisco IOS*, оптимизированная для организации телекоммуникаций, а также технология *Cisco Network Caching*, которая минимизирует избыточный трафик, передаваемый по каналам WAN. Она позволяет повысить производительность сети за счет того, что большинство запросов к внешним ресурсам исполняется локально, а не за счет передачи этих запросов к удаленным серверным группам. Такое решение защищает внутреннюю сеть от неконтролируемых перегрузок в интернете или в корпоративной сети, что позволяет повысить качество предоставляемых услуг и доступность информации, хранящейся на внешних серверах.

Архитектурно данная технология оптимизирована для использования одной кэширующей платформы, объединяющей в себе поддержку протокола WCCP на всех критичных устройствах активного сетевого оборудования. WCCP (Web Cache Control Protocol) - протокол, разработанный компанией *Cisco Systems* и ставший фактически стандартом для прозрачных кэширующих систем; он поддерживается практически всеми поставщиками устройств кэширования.

Cobalt Networks. Эта компания поставляет недорогие кэширующие продукты для предприятий малого и среднего бизнеса. В сентябре 2000 года она была приобретена компанией *Sun Microsystems* (*www.sun.ru*), поэтому новые продукты поставляются под маркой Sun Cobalt. Компания утверждает, что сокращение полосы частот, обеспечиваемое этими продуктами, достигает 50%.

Compaq Computer. В представленных в обзоре продуктах *Compaq*, образующих линейку для потребителей разных классов, используется ПО кэширования *Novell ICS Caching*. Однако *Compaq Computer Corporation* и *Inktomi* 21 марта 2001 года подписали соглашение, в соответствии с которым *Compaq* становится партнером *Inktomi* в разработке интегрированных платформ сетевой доставки контента (компания *Inktomi* специализируется на разработке ПО для масштабируемых сетевых инфраструктур для интернета). Целью указанного соглашения является разработка программных платформ и программно-аппаратных серверов для доставки контента. В рамках этой программы предполагается объединить кэш-серверы *Compaq TaskSmart C-Series* с ПО *Inktomi Traffic Server* и *Inktomi MediaBridge*. Такое изделие должно появиться на рынке летом 2001 года.

По утверждению компании используемая полоса уменьшается на величину до 30%, сокращение времени доставки контента пользователю (или провайдеру) или времени отклика сети достигает десятикратного, Разгрузка Web-серверов составляет до 80% запросов.

Intel. Для изделий этой корпорации характерна высокая отказоустойчивость. Имеется порт аварийного управления *Emergency Management Port* (EMP), позволяющий управлять устройством даже при отказе программного обеспечения или сети. Предусмотрено

несколько вариантов прозрачного режима, включая коммутацию четвертого уровня, WPAD и маршрутизацию на основе правил. Применяется архитектура *DataFlow*, которая обеспечивает одновременную буферизацию и передачу потоков данных для повышения пропускной способности и бесперебойного ввода/вывода.

Факторы, на которые необходимо обратить внимание при покупке

Технические характеристики. Одна из основных характеристик прокси-кэш-сервера - емкость дисковых массивов, используемых для кэширования, и эта характеристика представляется наиболее объективной и легко проверяемой. Что касается остальных характеристик, а именно экономии полосы пропускания, ускорения обращения к контенту и разгрузки Web-серверов, то они обычно оказываются трудноизмеримыми на практике, во всяком случае, такую проверку невозможно сделать при приобретении изделия, и к этим показателям, декларируемым производителями и продавцами, следует относиться скептически. Однако чем больше емкость и быстродействие дискового массива, тем, как правило, выше и остальные показатели.

Простота установки и использования. Наиболее предпочтительными с этой точки зрения являются приборы типа solution in a box, которые включают все необходимые аппаратные и программные средства и относятся к категории plug-and-play. Это, например, прокси-кэш-серверы CacheFlow и Intel.

Возможность изменения оптимизируемой характеристики. Основное предназначение прокси-кэш-сервера - экономия полосы пропускания и сокращение времени реакции на запросы пользователя. Опыт показывает, что достичь наибольшего выигрыша одновременно по этим двум показателям не удается и надо выбрать что-то одно, причем выбор того или иного варианта определяется конкретной задачей пользователя. По этой причине весьма желательно иметь возможность переключать режимы работы прокси-кэш- сервера: максимальная экономия полосы пропускания и максимальное

сокращение времени реакции. Например, в прокси-кэш-сервере CacheFlow просто имеется соответствующий тумблер.

Гибкость конфигурирования. С течением времени у пользователя может возникнуть необходимость в переконфигурировании прокси-кэш-сервера. В этом случае желательно, чтобы он был способен работать в соответствии с моделями прямого, обратного и прозрачного кэширования.

Администрирование плохо сконструированного прокси-кэш-сервера отнимает массу времени у системных администраторов, особенно если в корпорации установлен кластер таких серверов. Желательно, чтобы все прокси-кэш-серверы могли администрироваться централизованно и при этом имели бы интерфейсы на основе браузера и командной строки.

Масштабируемость и надежность. Высокая масштабируемость достигается в том случае, если возможна кластеризация прокси-кэш-серверов или построение иерархических кластерных систем. Базовым показателем надежности является коэффициент готовности, на который надо обращать особое внимание при покупке.

Размеры, масса, энергопотребление. Прокси-кэш-серверы могут существенно различаться по этим показателям, поэтому при их выборе следует учитывать возможности, предоставляемые тем рабочим местом, куда кэш-сервер предполагается установить.

Обновление информации в кэше

Современные кэши используют пассивное или активное кэширование. При пассивном кэшировании кэш-сервер проверяет свежесть контента. Обычно кэш-сервер посылает команду *get* (в HTTP) для запрашивания объекта от контент-сервера. В тех случаях, когда объект уже был сохранен, кэш-сервер использует модифицированную команду *get if*, в соответствии с которой объект скачивается, если он был изменен после последнего запроса. Затем кэш-сервер сравнивает даты изменения объекта, поступившего от сервера, и объекта, хранящегося в кэше, и направляет пользователю самый последний вариант.

Слабое место пассивного кэширования - производительность: пользователи должны ждать, пока кэш-сервер проверит каждый запрос. Однако в этом случае возможны некоторые улучшения, например, если производится проверка свежести данных по дате окончание заданного срока, заносимой в заголовок объекта. Когда объект достигает определенного «возраста», кэш-сервер запрашивает свежий контент.

При активном кэшировании улучшение характеристик достигается с использованием эвристических методов оценки срока жизни объекта. Сервер при этом проводит вычисления, используя такие данные, как дата занесения объекта в кэш, продолжительность его пребывания в кэше, IP-адрес источника и множество подобных сведений. При таком подходе не нужно проверять каждый запрос. Вместо этого кэш- сервер делает определенные предположения о времени жизни объекта, скажем два дня. В течение этого интервала все запросы объекта немедленно обслуживаются из кэша, однако по истечении этого срока кэш обновляет объект.

Размещение прокси-кэш-серверов

Различают логическое и физическое размещение прокси-кэш-сервера. В случае корпоративной сети используют три основные логические конфигурации:

- сервер помещается рядом с маршрутизатором (в этом случае он обрабатывает трафик по протоколу управления Web-кэшированием, WCCP);
- сервер объединяется с коммутатором четвертого или более высокого уровня (в этом случае он управляет трафиком);
- сервер встраивается в коммутатор второго или третьего уровня.

Физически прокси-кэш-сервер необходимо размещать как можно ближе к пользователям. Для корпоративной сети это означает, что кэширование должно осуществляться на границе сети.

Что касается сотрудников корпорации, работающих на дому, то кэширование нужной им информации выполняют прокси-кэш-серверы, размещенные в точке присутствия ISP.

УСТАНОВКА И НАСТРОЙКА ПРОКСИ-СЕРВЕРА

Подготовка к установке SQUID.

В первую очередь нам необходимо скачать прокси сервер. В данном примере мы разместим исходники сервера в разделе /root/distr. Так, скорее всего, его в вашей системе нет, то следующим блоком команд мы его создадим и загрузим туда SQUID.

Сперва необходима скачать дистрибутив из интернета и скинуть его на флешку. Затем монтируем флешку:

mount –t msdosfs /dev/da0s1 /mnt

Затем создадим директорию, куда будем копировать:

freebsd# mkdir -p /root/distr

Теперь скопируем архив в нашу папку:

freebsd# cp /mnt/squid-2.6.STABLE7-20070120.tar /root/distr

После скачивания распаковываем архив:

freebsd# tar xvfz squid-2.6.STABLE7-20070120.tar.gz freebsd# cd squid-2.6.STABLE7-20070120.tar

Теперь все готово к компиляции и установке!

Компиляция и установка прокси сервера SQUID

Скомпилировать и установить довольно просто. Выполняем следующий набор команд:

 # ./configure --prefix=/usr/local/squid # make all
 # make install

Если команда configure выдаст ошибку из за отсутствия в системе Perl вы можете легко это исправить добавив его в систему следующей командой:

 # pkg_add -r perl

Или же установив дистрибутив, скачав его из интернета и скопировав его на ваш компьютер с флешки. Чтобы установить дистрибутив, необходимо разархивировать его и выполнить следующие команды:

sh Configure -de make
make test make install

После добавления Perl повторите набор команд конфигурации и установки.

Настройка SQUID

Файл конфигурации лежит в следующей директории

/usr/local/squid/squid.conf

Открываем его любым редактором и первое что необходимо задать - "visible_hostname". Например:

visible_hostname freebsd

Замените freebsd на любое имя хоста и сохраните файл (необходимо, чтобы имя хоста совпадало с именем вашего компьютера, на котором вы устанавливаете SQUID).

Следующее что мы должны сделать для запуска squid это создать раздел в который будем сохранять логи и где squid будет хранить cache. И конечно настроить доступ и разрешения к этим разделам. Выполним следующий набор команд:

 # mkdir -p /usr/local/squid/var/logs/ # chmod 777 /usr/local/squid/var/logs/
 # mkdir -p /usr/local/squid/var/cache/ # chmod 777 /usr/local/squid/var/cache/

И обязательно необходимо что бы squid создал структуру разделов для хранения cache перед его первым запуском. Выполнить эту команду обазятельно:

 # /usr/local/squid/sbin/squid –z

Теперь squid готов к первому запуску!

Старт и остановка прокси сервера

Для того что бы запустить squid достаточно выполнить следующую команду:

 # /usr/local/squid/sbin/squid

Остановить squid можно так:

 # kill -9 `cat /usr/local/squid/var/logs/squid.pid`

Можно настроить запуск SQUID при старте системы создав следу ющий файл /usr/local/etc/rc.d/squid.sh с минимальным содержанием:

#!/bin/sh
/usr/local/squid/sbin/squid

Обязательно надо разрешить выполнять данный файл:

 # chmod 755 /usr/local/rc.d/squid.sh

Конфигурационный файл

```
  visible_hostname anyhost http_port 8080
  icp_port 3130
  cache_peer 10.5.2.24 parent 8080 3130 proxy-only
  acl QUERY urlpath_regex cgi-bin \?
  no_cache deny QUERY
  dns_nameservers 10.5.2.24
  auth_param basic children 5
  auth_param basic realm Squid proxy-caching web server
  auth_param basic credentialsttl 2 hours

  refresh_patte   ^ftp:      1440    20% 10080
rn
  refresh_patte   ^gopher:   1440    0%  1440
rn
  refresh_patte   .   0      20%     4320
rn
  acl all src 0.0.0.0/0.0.0.0
  acl manager proto cache_object
  acl localhost src 127.0.0.1/255.255.255.255
  acl to_localhost dst 127.0.0.0/8
  acl SSL_ports port 443 563
  acl Safe_ports port 80      # http
  acl Safe_ports port 21      # ftp
  acl Safe_ports port 443 563 # https, snews
  acl Safe_ports port 70      # gopher
  acl Safe_ports port 210     # wais
  acl Safe_ports port 1025-65535   # unregistered ports
  acl Safe_ports port 280     # http-mgmt
  acl Safe_ports port 488     # gss-http
  acl Safe_ports port 591     # filemaker
  acl Safe_ports port 777     # multiling http
  acl CONNECT method CONNECT
  acl get_post method GET POST


  acl net2 src 10.5.2.0/255.255.255.0
  acl net219 src 10.5.219.0/255.255.255.0
  acl net220 src 10.5.220.0/255.255.255.0
  acl net224 src 10.5.224.0/255.255.255.0
  acl net226 src 10.5.226.0/255.255.255.0
  acl net231 src 10.5.231.0/255.255.255.0
  acl net157 src 10.5.157.0/255.255.255.0
```

```
acl net158 src 10.5.158.0/255.255.255.0
http_access allow net2
http_access allow net219
http_access allow net220
http_access allow net224
http_access allow net157
http_access allow net158
http_access allow net226
http_access deny net231
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_reply_access allow all icp_access allow all
uri_whitespace encode nonhierarchical_direct off
coredump_dir /usr/local/squid/var/cache
 request_entities on
```

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Под руководством преподавателя самостоятельно настроить прокси-сервер Squid и проверить его работоспособность. Для этого необходимо:

1. Скачать дистрибутив из интернета и скинуть его на флешку.
2. Смонтировать флешку.
3. Создать директорию, в которую будем копировать дистрибутив.
4. Скопировать архив в созданную директорию и распаковать его.
5. Скомпилировать и установить дистрибутив.
6. Настроить файл конфигурации squid.conf согласно примеру.
7. Запустить прокси-сервер.
8. Изменить права у файла squid.sh.
9. Проверить работоспособность прокси-сервера.
10. Завершить работу FreeBSD.

Ответить на контрольные вопросы и подготовить отчет.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Дайте определение прокси-сервера.
2. Укажите цели применения прокси-сервера.
3. Опишите преимущества в работе в сети Интернет с использованием прокси.
4. Перечислите виды прокси.
5. Дайте определение HTTP-прокси и перечислите его возможности.
6. Дайте определение FTP-прокси.
7. Дайте определение HTPS-прокси и укажите его отличия от HTTP-прокси.
8. Дайте определение Mapping-прокси.
9. Дайте определение Socks-прокси.
10. Опишите принцип кэширования.
11. Опишите принцип сетевых кэшей и их назначение.
12. Дайте определение ICP и HTCP.
13. Укажите отличия кэш-сервера от прокси-сервера.
14. Опишите назначение кэш-сервера.
15. Дайте определение прокси-кэш-сервера и опишите его концепцию.
16. Опишите принцип прозрачного кэширования.
17. Перечислить архитектуры(модели) прокси-кэш-сервера и описать их суть.
18. Перечислите фирмы, занимающиеся разработкой и производством прокси-кэш-сервером и охарактеризуйте их продукцию.
19. Перечислите детали, на которые стоит обратить внимание при покупке прокси-кэш-сервера.
20. Опишите принцип активного и пассивного кэширования и их отличия.
21. Опишите принцип каскадной настройки прокси серверов.

ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 3 занятия (6 академических часов: 5 часов на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета).

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания, ответы на контрольные вопросы, описание процесса выполнения лабораторной работы, выводы.

ОСНОВНАЯ ЛИТЕРАТУРА

1. Вирт, Н. Разработка операционной системы и компилятора. Проект Оберон [Электронный ресурс] / Н. Вирт, Ю. Гуткнехт. — Москва: ДМК Пресс, 2012. 560 с. Режим доступа: https://e.lanbook.com/book/39992

2. Войтов, Н.М. Основы работы с Linux. Учебный курс [Электронный ресурс]: учебное пособие / Н.М. Войтов. — Москва : ДМК Пресс, 2010. — 216 с. — Режим доступа: URL: https://e.lanbook.com/book/1198

3. Стащук, П.В. Краткое введение в операционные системы [Электронный ресурс] : учебное пособие / П.В. Стащук. — 3-е изд., стер. — Москва : ФЛИНТА, 2019. — 124 с.— URL: https://e.lanbook.com/book/125385

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

4. Войтов, Н.М. Администрирование ОС Red Hat Enterprise Linux. Учебный курс [Электронный ресурс] : учеб. пособие — Москва: ДМК Пресс, 2011. 192 с. Режим доступа: https://e.lanbook.com/book/1081

5. Стащук П.В. Администрирование и безопасность рабочих станций под управлением Mandriva Linux: лабораторный практикум. [Электронный ресурс]: учебно-методическое пособие / П.В. Стащук. — 2-е изд., стер. - М: Флинта, 2015. https://e.lanbook.com/book/70397

Электронные ресурсы:

1. Научная электронная библиотека http://eLIBRARY.RU.
2. Электронно-библиотечная система http://e.lanbook.com.
3. Электронно-библиотечная система «Университетская библиотека онлайн» http://biblioclub.ru.
4. Электронно-библиотечная система IPRBook http://www.iprbookshop.ru/
5. Losst - Linux Open Source Software Technologies https://losst.ru