СИСТЕМА МОНИТОРИНГА ИНТЕРНЕТ ТРАФИКА

Пающик Ю.В., Родригес Залепинос Р.А.
Донецкий национальный технический университет
кафедра компьютерных систем мониторинга
paushcik.julia@mail.ru

Рассмотрены методы и способы анализа содержимого Интернет трафика. Продемонстрирована архитектура системы анализа содержимого Интернет трафика, предусматривающая внедрение прозрачной очистки трафика перед его просмотром пользователями. Реализована подсистема доставки содержимого Интернет трафика множеству одновременных пользователей в реальном времени.

Введение

На сегодняшний день Интернет является активной средой распространения и сбора информации. С помощью интеллектуального анализа данных можно автоматически обнаружить ранее неизвестные полезные знания в Интернет страницах. Коммерческие Интернет страницы, как правило, содержат много блоков информации. Помимо блоков основного содержания, на странице также находятся блоки панелей навигации, уведомления об авторском праве и конфиденциальности и рекламные объявления. В интеллектуальном анализе Интернет страниц такие блоки называют шумными блоками (noisy blocks).

Доставка Интернет страниц на мобильные устройства стало возможным с появлением беспроводных технологий. Однако, у мобильных устройств маленькие экраны и объемы памяти. Преобразование Интернет страниц для поставки на мобильные устройства является важной проблемой. Существует масса алгоритмов по очистки «шумных блоков», чтобы оценить объекты контента в пределах Интернет страницы. Это позволяет выбрать только важные части Интернет страницы для доставки на мобильные устройства. Помимо этого устранение шумных блоков может улучшить анализ содержимого Интернет страниц. Также важно устранять шумные блоки для улучшения восприятия информации пользователем мобильного устройства.

С целью обеспечения прозрачной очистки Интернет страниц разработана архитектура, которая вводит программный шлюз-посредник между клиентской машиной и запрашиваемой Интернет страницей. Он функционирует в режиме обычного прокси сервера. Браузеры на клиентских машинах настраиваются на работу с разработанным прокси сервером. При запросе необходимой Интернет страницы, разработанный прокси сервер анализирует URL и загружает себе заданную страницу. На данный момент шлюз настроен на трансляцию неизменной страницы клиентским браузерам. Архитектура и реализация программного шлюза предусматривает внедрение методов очистки страниц.

Программный шлюз реализован на языке Java и использованием неблокирующих сокетов. Особенностью шлюза является поддержка одновременного подключения тысяч клиентов и передачу содержимого страниц в реальном времени. Под передачей

данных в реальном времени понимается загрузка Интернет страниц для множества одновременных пользователей без заметной для них задержки, связанной с внедрением сервера-посредника.

Архитектура системы анализа содержимого Интернет страниц

Программный шлюз является посредником между компьютером пользователя и запрашиваемой Интернет страницей. Пользователь вводит нужный ему URL в адресную строку браузера, у которого в настройках параметров соединения с Интернетом указано подключение через разработанный прокси сервер, после чего шлюз-посредник выполняет запрос по указанному URL адресу. Программный шлюз получает содержимое Интернет страницы по заданному URL и обрабатывает его в реальном времени. Система анализа содержимого предусматривает загрузку Интернет страниц для множества одновременных пользователей. В результате клиент получает содержимое запрашиваемой Интернет страницы, отображаемой браузером. Наглядная демонстрация передачи обработанных Интернет страниц через шлюз в реальном времени множеству одновременных пользователей (рис. 1).

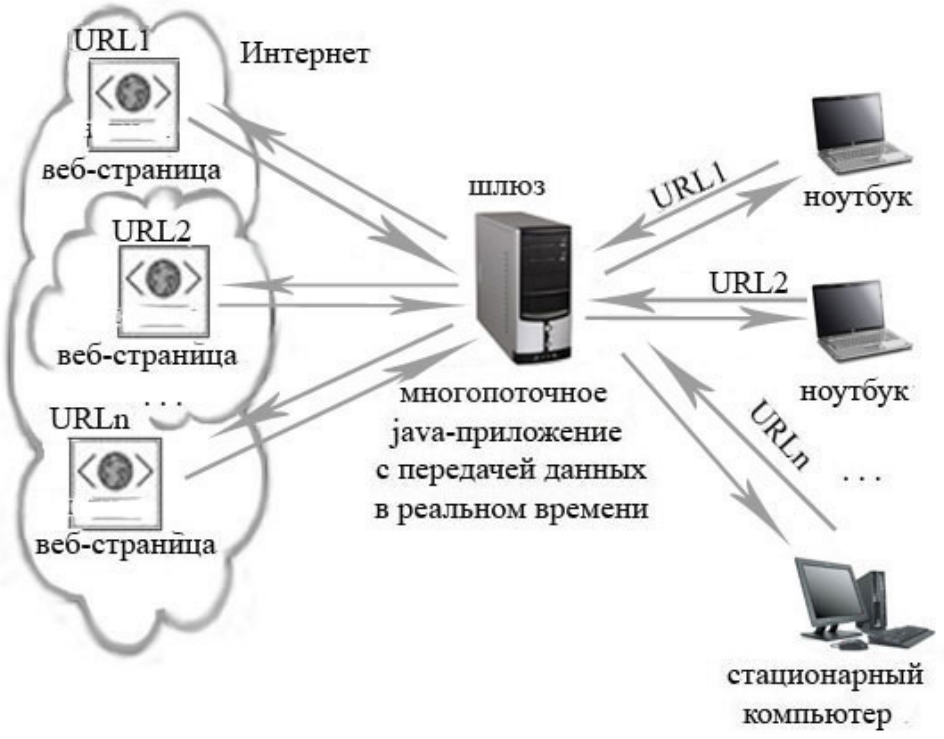

Рисунок 1. Наглядная демонстрация передачи обработанного Интернет трафика через шлюз множеству одновременных пользователей в реальном времени

Программный шлюз анализа содержимого Интернет страниц

Программный шлюз реализован на кроссплатформенном языке программирования Java. Это обеспечивает работоспособность шлюза более чем на одной аппаратной платформе и/или операционной системе.

Разработанный программный шлюз является многопоточным приложением. Преимущество многопоточности заключается в обеспечении наиболее эффективной реализации процедуры квантования времени. (Код переключения между задачами

на Java выглядел бы куда более громоздко, чем независимое описание действий для каждого потока). Программный шлюз предназначен для задач требуемых выполнения нескольких действий одновременно, что подразумевает использование многопоточности. Прокси серверу общего пользования необходимо обслуживать несколько клиентов одновременно. Когда имеется несколько потоков клиентов и необходимо обслуживать их все одновременно необходима реализация многопоточности для удобства программной реализации. Это позволяет работать шлюзу в реальном времени для множества одновременных пользователей.

Фундаментом реализации программного шлюза является библиотека MINA. MINA – акроним для «Multipurpose Infrastructure for Network Applications» («Многоцелевая инфраструктура для сетевых приложений»). В настоящее время MINA поддерживает протоколы TCP и UDP на основе Java NIO API, обеспечивает поддержку для последовательного порта, и транспортирует на основе выполнения Apache Portable. На данный момент вышло 3 версии Мины 1.0, 1.1, 2.0. Разработчики советуют использовать последнюю точку-релиза 2.0 (для Java 5 или выше). Для использования MINA требуется JDK 1.5 или выше. Но MINA работает идеально и с JDK 1.4, только если не использовать SSLFilter, который использует Java 5 SSLEngine. Это означает, JDK 1.5 или выше требуется для использования SSL с MINA. SSL (*Secure Sockets Layer* – уровень защищённых сокетов) – криптографический протокол, который обеспечивает установление безопасного соединения между клиентом и сервером.

Основной модуль MINA зависит от двух библиотек, SLF4J и backport-util-concurrent (для 1.0). SLF4J (Simple Logging Facade for Java) предоставляет привязки для Log4J, JDK 1.4 API регистрации и NLog4J. Backport-util-concurrent является реализацией портирования из пакета `java.util.concurrent`, который был введен в JDK 1.5. MINA 1.0 использует это, чтобы сохранить время выполнения совместимости с JDK 1.4. MINA 1.1 или выше, зависит от пакет `java.util.concurrent` напрямую, и поэтому не зависят от него больше.

MINA позволяет достаточно легко создавать клиентские и серверные приложения, используя классы `IoConnector` и `IoAcceptor`.

При реализации программного шлюза использовалась структура MINA. Учитывая правила реализации MINA для программного шлюза изначально необходим объект для прослушивания входящих соединений. Так как шлюз должен поддерживать протоколы TCP/IP, то в роли объекта будет выступает acceptor класса NioSocketAcceptor. После чего нужна реализация конфигураций фильтров, обрабатывающих информацию. Первым добавляется фильтр для перевода двоичных или протокольных данных в объект сообщения. За это отвечает фильтр ProtocolCodecFilter. Затем необходим фильтр для хранения списка всех действий, а именно вновь созданных сессий, сообщений получателя и отправителя, и т.д. Фильтр выполняющий эти действия – LoggingFilter. После установки конфигурации фильтров определяется обработчик, который используется для обслуживания клиентских подключений и запросов на текущее время. Созданный класс ServerHandler является обработчиком, так как реализует интерфейс IoHandler. Этот обработчик является главным классом проекта помимо класса Server, содержащего метод main, а значит будет обслуживать все входящие запросы от клиента.

Обязательным этапом является инициализация порта, который прослушивается шлюзом, и указания какие действия будут возможны при открытой сессии. С помощью класса InetSocketAddress определяется какой хост будет прослушиваться, а методом bind устанавливается, что при открытой сессии можно читать данные запроса и в то же время отправлять ответ клиентам. Наглядно показано использование структуры MINA в создании шлюза (рис. 2).

Рисунок 2. Демонстрация реализации конфигураций работы шлюза, используя структуру MINA

Алгоритмы разбора запросов для реализации программного шлюза

Программный шлюз обрабатывает запрос URL от пользователя, разбивает его по параметрам, парсит. Распасенный запрос заполняет список Map по параметрам URI, Method, Context, Protocol. Запрос GET передает данные серверу используя URL. Схема кодирования URL для метода пересылки GET заключается в следующем. Для каждого элемента формы, имеющего имя, формируется пара «*name=value»,* где *value* – значение элемента, введенное пользователем или назначенное по умолчанию. Значение может отсутствовать. Все пары объединяются в строку, в качестве разделителя служит символ «&». Так как имена и значения представляют собой обычный текст, то они могут содержать символы, недопустимые в составе URL. Такие символы заменяются последовательностью, состоящей из символа «%» и их шестнадцатеричного ASCII-кода. Символ пробела может заменяться не только кодом «%20», но и знаком «+».

При чтении запроса класс HttpRequestDecoder учитывает символы URL запроса при разбиении метода GET. Класс HttpRequestMessage служит для временного хранения распарсенного запроса и предоставляет доступ к частям запроса. Класс ServerHandler является обработчиком, который используется для обслуживания клиентских подключений и запросов на текущее время. В нем определяются действия во время простоя сессии и время, в течении которого сессия будет простаивать. Так же этот обработчик отвечает за извлечение типа протокола HTTP. Метод doHttpUrlConnectionAction принимает параметр домена, который запрашивает пользователь, и создает объект класса HttpURLConnection, из которого извлекается тип протокола HTTP. Это позволяет получить содержимое Интернет страницы. В классе HttpResponseEncoder формируется ответ клиенту на его запрос, учитывая стандарты формирования кодировки ответа. В классе HttpResponseMessage хранится тело HTTP ответа клиенту и осуществляется коннект вывода в браузер Интернет страницы, обработанной методом doHttpUrlConnectionAction класса ServerHandler. Диаграмма реализованных классов (рис. 3).

На данном этапе шлюз настроен на трансляцию неизменной страницы клиентским браузерам. Архитектура и реализация программного шлюза предусматривает внедрение методов очистки страниц от «шумных блоков» с целью улучшения анализ данных Интернет страниц.

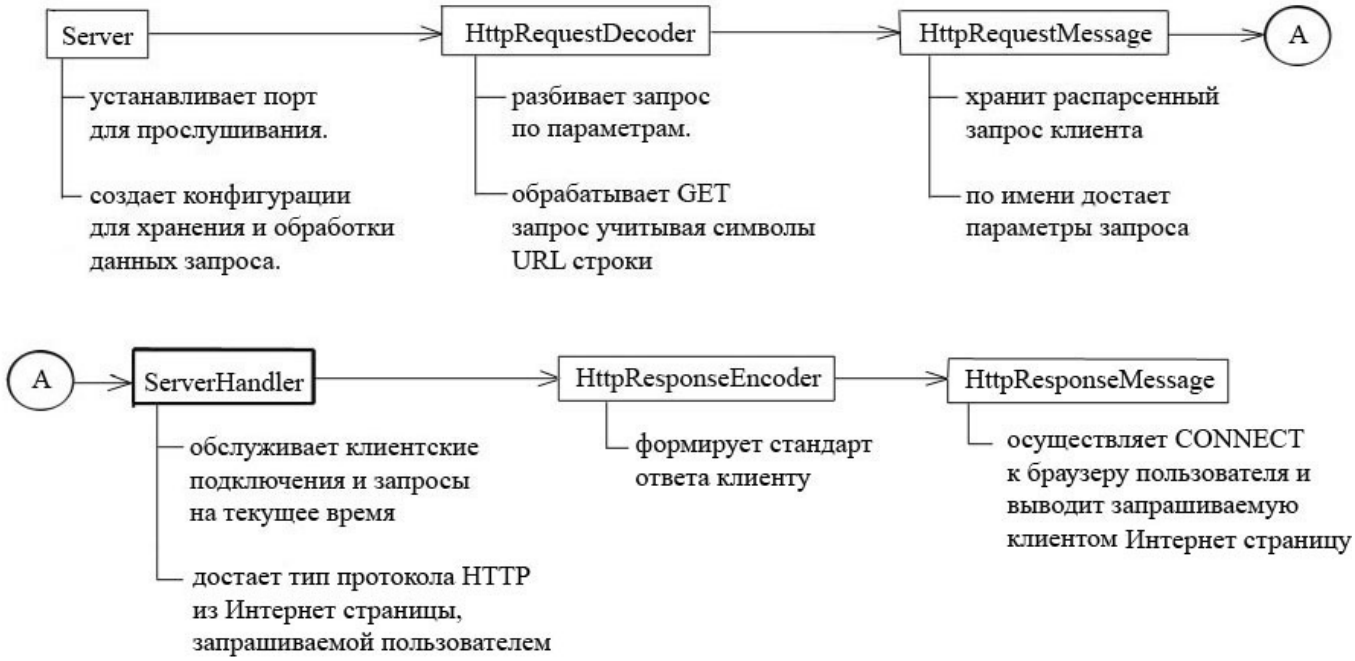

Рисунок 3. Наглядная диаграмма работы классов программного шлюза

Выводы

Были рассмотрены методы анализа содержимого Интернет страниц. Предложена архитектура системы анализа содержимого Интернет страниц, предусматривающая в прозрачную очистку Интернет страниц перед их просмотром пользователями. Очистка Интернет страниц улучшает восприятие их содержимого пользователями мобильных устройств. Реализована подсистема доставки содержимого Интернет страниц множеству одновременных пользователей в реальном времени.

Литература

[1] MINA [Электронный ресурс] – Режим доступа: http://mina.apache.org/

[2] URLConnection [Электронный ресурс] – Режим доступа: http://docs.oracle.com/javase/tutorial/networking/urls/readingWriting.html

[3] Chakrabarti, S. Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufmann, 2002.

[4] Anderberg, M.R. Cluster Analysis for Applications, Academic Press, Inc. New York, 1973.

[5] Lee, M.L., Ling, W. and Low, W.L. Intelliclean: A knowledge-based intelligent data cleaner. KDD-2000, 2000.

[6] Nahm, U.Y., Bilenko, M. and Mooney R.J. Two Approaches to Handling Noisy Variation in Text Mining. ICML-2002 Workshop on Text Learning, 2002.