УДК 004.738:004.056

Д.О. Тарасов, С.В. Симоненко
Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж

ІНТЕЛЕКТУАЛЬНИЙ СЕРВІС БАГАТОКОРИСТУВАЦЬКОГО ДОСТУПУ ДО ІНТЕРНЕТ

Розглянуто питання створення інтелектуального сервісу багатокористувацького доступу до Інтернет. Основними завданнями інтелектуального сервісу багатокористувацького доступу є забезпечення анонімної роботи користувачів локальної мережі з Web-сайтами, фільтрації HTTP контента, облік та оптимізація використання трафіка.

The research considers the problems of providing multiuser access to the Internet. The main purposes of article is a providing of anonymous work for local net users of web sites, filtering of HTTP content, traffic usage calculation and optimization.

Постановка проблеми у загальному вигляді

Користуючись сучасними веб-браузерами, користувачі, крім запитів на конкретні веб-документи, відправляють до веб-серверів також багато іншої інформації, вміст якої інколи може бути конфіденційним або заважати нормальній роботі у Інтернет.

Деякі Інтернет-компанії за допомогою так званих cookies збирають інформацію про користувача та використовують її у власних, не завжди відомих користувачу цілях. Крім того, cookies можуть містити конфіденційну інформацію – паролі на інші веб-сайти тощо [4–6].

Анонімізацією роботи в Інтернет називають комплекс заходів, які мінімізують передавання в мережу Інтернет персональної інформації користувача та інформації, яка може сприяти визначенню особи, характеру, поведінки користувача.

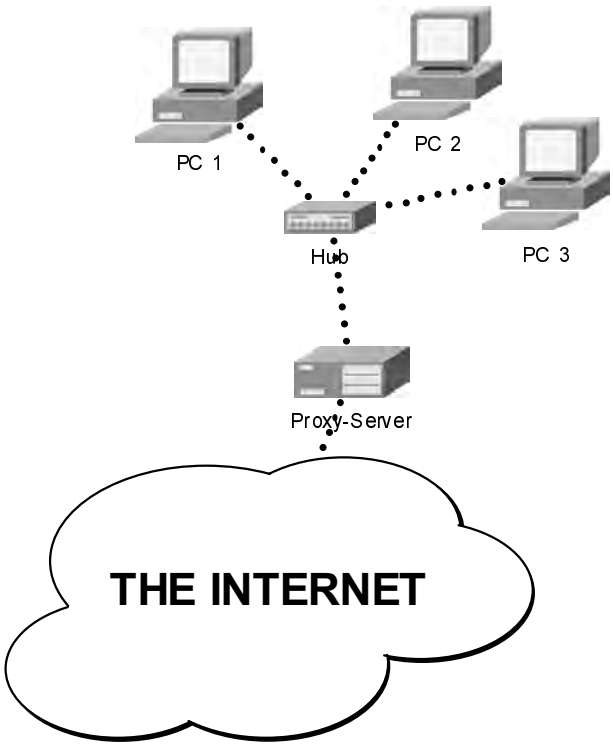

Рис. 1. Схематичне зображення доступу комп'ютерів до зовнішньої мережі

Анонімізація дозволяє заборонити роботу з cookies для окремих веб-сайтів і тим захищає користувача від вищевказаних незручностей. Крім того, анонімізація дозволяє повністю закрити інформацію про веб-браузер та ОС користувача, що інколи буває доцільно, оскільки деякі веб-сайти, незважаючи на підтримку веб-браузерами Netscape та Opera 6-го покоління всіх сучасних стандартів, потребують для роботи тільки Internet Explorer. Власники веб-сайтів можуть заборонити доступ окремим користувачам до частини чи до всіх ресурсів або відслідкувати користувача, що звернувся до веб-ресурсу та, знаючи його IP-адресу та параметри системи, заподіяти шкоду.

Анонімізація дозволяє відправляти всі запити назовні з визначеними користувачем параметрами, тим самим ховаючі IP-адресу та системні дані користувача [3].

Прозорість запитів доцільна для роботи з локальної мережі із внутрішньою адресацією. На відміну від анонімізації, прозорість забезпечує повне збереження структури та даних у запитах до зовнішніх веб-серверів, надаючи тим самим користувачам локальної мережі повноцінний доступ до веб-ресурсів. Схематичне зображення доступу локальних комп'ютерів до зовнішньої мережі зображено на рис. 1.

Цілі статті

Дослідивши предметну галузь та розлянувши основні потоки даних, які характеризують цю задачу, було побудовано відповідну діаграму потоків даних предметної галузі (рис. 2).

Рис. 2. Діаагрма потоків даних інтелектуального сервісу фільтрації контента

Клієнт передає запит на proxy-сервер за протоколом HTTP, після чого proxy-сервер проводить інтелектуальний аналіз запиту, фільтрацію заголовків, перенаправляє його на батьківський proxy-сервер (parent-proxy) або сам формує відповідь і повертає її назад.

Додатковою функцією інтелектуального сервісу багатокористувацького доступу до Інтернет є маршрутизація вихідного HTTP трафіка з метою пришвидшення доступу та мінімізації витрат.

Для мінімізації витрат та пришвидшення отримання Web-сторінок HTTP запити клієнтів передаються кеш-серверам або батьківському proxy-серверу, який виконає запит клієнта за найнижчими тарифами зв'язку (або найшвидше). Для обрання кешуючого або батьківського серверу інтелектуальний сервіс використовує базу правил маршрутизації HTTP трафіка.

Аналіз останніх досліджень

Серед існуючих систем фільтрації контента, що передається за протоколом http, для дослідження обрано 4 найбільш поширених:

- Proximitron, автор Scott R. Lemmon. http://proxn.btnet.org ;
- SquidGuard, Squid Software. http://www.squidguard.org ;
- JesRed, автор Denis Shaposhnikov. http://ivs.cs.uni-magdeburg.de/~elkner/webtools/jesred/ ;
- Apache з модулем mod_rewrite, Apache Software Foundation. http://httpd.apache.org/ .

Результати досліджень перелічених вище систем наведено у таблиці.

Дослідження існуючих систем фільтрації контента

Система	Платформа	Переваги	Недоліки
Proximitron	Windows	• малий об'єм • висока швидкість роботи • велика вбудована база правил	• недостатя масштабованість • некросплатформна • працює з одним parent-proxy
SquidGuard	UNIX, Windows	• висока масштабованість правил	• працює як зовнішній фільтр для Squid • показує на Windows-платформі погані результати за швидкодією та надійністю • праює з одним parent-proxy
JesRed	UNIX, Windows	• малий об'єм	• недостатя масштабованість • аналогічно до SquidGaurd працює тільки із Squid • працює з одним parent-proxy
Apache з модулем mod_rewrite	UNIX, Windows	• достатня масштабованість правил	• великий об'єм • відсутні деякі функції • слабкий захист • працює з одним parent-proxy

Основний матеріал

Забезпечення інтелектуальної фільтрації контента

Важливою функцією інтелектуального сервісу багатокористувацького доступу до Інтернет є фільтрація вхідної та вихідної інформації (контента). Функція фільтрації контента потрібна для блокування доступу до окремих сайтів та доменів, економії трафіка за рахунок блокування реклами (банерів), створення більш комфортних умов роботи користувачів.

Для забезпечення інтелектуальної фільтрації сервіс багатокористувацького доступу перехоплює такі параметри HTTP-запиту, як заголовок, хост та деякі інші.

За допомогою масок, регулярних виразів та часових змінних інтелектуальний сервіс вибирає та опрацьовує запити відповідно до заданих правил.

Наприклад:

Маска запиту 1 → Правило 1 – заблокувати ресурс

Маска запиту 2

Маска запиту 3 → Правило 2 – заблокувати ресурс та видати деякий локальний документ

Маска запиту 4 → Правило 3 – змінити заголовок запиту та забезпечити анонімність

Маска запиту 5 → Правило 4 – передати запит на веб-сервер та змінити отриманий документ.

Правила зберігаються у SQL базі даних, що полегшує внесення змін та гнучкість систем.

Протоколи інтелектуального сервісу

Базовим протоколом, який підтримує інтелектуальний сервіс, обрано HTTP. Протокол HTTP (Hypertext Transfer Protocol, протокол передачі гіпертекстів) [4, 5] — це протокол прикладного рівня для розподілених гіпертекстових інформаційних систем. Крім передачі гіпертекстів, він може застосовуватися і в інших галузях, таких, як сервери імен і розподілені системи управління об'єктами, але для даної роботи важливо те, що на цьому протоколі базується World Wide Web.

Було також проведено додаткові дослідження, які показали недоцільність використання систем фільтрації для таких сервісів, як POP, IMAP, SMTP, FTP, TELNET, IRC, оскільки:

- ці протоколи є достатньо простими порівняно із HTTP;
- жоден із перелічених вище протоколів стандартно не є пристосованим для проведення запитів до ресурсу через proxy-сервер;
- для всіх перелічених вище протоколів існують SSL-аналоги, які, на відміну від HTTPS, як правило, не потребують купівлі додаткового SSL-сертифіката та набули такого поширення, що у найближчі 2–3 роки, вони можуть повністю вивести з обігу не-SSL-аналоги цих протоколів. Оскільки через SSL-з'єднання проходить лише зашифрований трафік, фільтрацію проводити неможливо.

HTTP є протоколом типу запит/відповідь. Клієнт посилає серверу запит, що складається з типу запиту, URI і версії протоколу, за якими слідує повідомлення, що містить модифікатори запиту, клієнтську інформацію і, можливо, тіло запиту. Сервер відповідає рядком стану, що містить версію протоколу і код стану, за якою слідує повідомлення, що містить серверну інформацію, метаінформацію і, можливо, тіло повідомлення.

Насправді ситуація зазвичай є складнішою через участь в процесі обміну інформацією посередників між клієнтом і сервером. Існують три такі посередники: проксі-сервер, шлюз і тунель:

Проксі-сервер (або сервер повноважень) приймає запит клієнта, включаючи повний URI, переписує все повідомлення або частину його і передає виправлений запит серверу, вказаному в URI.

Шлюз розташовується над сервером і при необхідності транслює запити в протокол більш низького рівня, підтримуваний сервером.

Тунель не змінює передаваних повідомлень, а використовується при передачі даних через посередника типу брандмауера (firewall).

Клієнт і сервер взаємодіють через додатковий сервер повноважень – інтелектуальну компоненту багатокористувацкого доступу.

Повідомлення HTTP

Повідомлення HTTP складаються із запитів клієнта до сервера і відповідей сервера клієнту. Обидва типи повідомлень складаються з:

- стартового рядка (визначає тип повідомлення);
- полів заголовка;
- порожнього рядка, що містить тільки символи CR LF і вказує на кінець заголовків;
- необов'язкового тіла повідомлення.

Кожне з полів заголовків має формат:

Ім'я поля ":" значення поля

Імена полів не залежать від регістра. Порядок, в якому передаються поля заголовка, неважливий, хоча зазвичай першими йдуть загальні поля заголовка, потім заголовок запиту або відповіді, а потім заголовки тіла документа.

HTTP допускає об'єднання декількох полів заголовка, що мають одне ім'я. Таке поле заголовка складається з імені поля, двокрапки і набору значень, розділених комами:

ім'я поля ":" значення1 "," значення2

Якщо тіло повідомлення зашифроване, то частина полів заголовка може бути причеплена до нього, тобто додана в кінець тіла повідомлення і також зашифрована.

Рядок запиту має вигляд

метод " " URI запиту " " версія HHTP CR LF

Тут метод — ця назва типу запиту, URI запиту — це або URI запрошуваного ресурсу, або символ "*" в тих випадках, коли запит не пов'язаний з конкретним ресурсом, а версія HHTP виглядає так:

"HTTP/" цифра "." цифра

Типові приклади рядків запиту:

GET http://www.mysite.com/index.html HTTP/1.1

OPTIONS * HTTP/1.1

Запит OPTIONS

Метод OPTIONS є запитом інформації про умови комунікації, доступні в ланцюжку запит/відгук для даного URI запиту. Він дозволяє клієнту визначити опції і/або вимоги, пов'язані з даним ресурсом, не виконуючи жодних операцій з ресурсом.

Якщо URI запиту задано символом "*", то запит належить до сервера, а не до конкретного ресурсу, і дозволяє запитати можливості сервера.

Запит GET

Метод GET використовується для запиту конкретного ресурсу. Цей метод є умовним, якщо в його заголовку задані поля If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match або If-Range. Умовний метод відрізняється тим, що ресурс, який було запитано, буде повернений тільки у тому випадку, коли він задовольняє задану умову.

Запит HEAD

Метод HEAD повністю аналогічний методу GET, але за ним сервер повертає тільки метаінформацію, пов'язану з документом, який було запитано, а не повністю весь документ Зазвичай цей метод використовується клієнтами, які підтримують кешування ресурсів, для перевірки того, чи слід відновити інформацію в кеші і перезавантажити документ.

Запит POST

Метод POST використовується в таких випадках:

- створення анотацій до наявних ресурсів;
- посилання повідомлення на дошку оголошень, в групу новин, список поштової розсилки і т. п.;
- передача блоку даних, наприклад, результату заповнення форми, програмі обробки даних;
- додавання записів в базу даних.

Суть даного методу полягає в тому, що тіло запиту передається серверу, і той інтерпретує його залежно від URI запиту (переважно це адреса програми, яка повинна обробити дані, що містяться в тілі запиту). Наприклад, якщо URI запиту містить ім'я Cgi-сценарію, то цей сценарій буде запущений на сервері і як параметри стандартного входу йому буде передано тіло запиту POST.

Запит PUT

Метод PUT використовується для створення нового ресурсу із заданим в запиті URI. Якщо ресурс з таким URI вже існує, то тіло запиту розглядається як його оновлена версія.

Запит DELETE

Метод DELETE використовується для видалення ресурсу, що має заданий URI.

Запит TRACE

Метод TRACE використовується в цілях відлагодження і діагностики. Він полягає в посиланні повідомлення заданим ланцюжком.

Запит CONNECT

Ім'я цього методу зарезервовано для проксі-серверів, які можуть динамічно перемикатися в тунельний режим.

Формат відгуку

Після отримання запиту та його інтерпретації сервер відправляє клієнту відгук такого формату:

- рядок стану;
- заголовки;
- CR LF;
- тіло повідомлення.

Рядок стану має вигляд:

версія HTTP " " код стану " " опис CR LF

Тут версія HTTP має той самий формат, що і в запиті, коди стану описані нижче, а опис — це рядок, що містить короткий текстовий опис кода стану.

Коди стану

Код стану — це ціле число з трьох цифр, яке вказує на результат виконання запиту. Перша цифра цього числа визначає клас відгуку:

- Інформаційні повідомлення (запит отриманий, йде його обробка).
- Запит оброблено успішно (запит був отриманий та зрозумілий сервером).
- Перенаправлення (для завершення запиту необхідні подальші дії клієнта).
- Помилка клієнта (запит неправильно складений або не може бути виконаний).
- Помилка сервера (сервер не зміг виконати запит за внутрішніми причинами).

Дві цифри коду стану, що залишилися, визначають його конкретне значення у межах даного класу.

Програмна реалізація інтелектуального сервісу

Вибір засобів реалізації

Сьогодні одним із найбільш оптимальних вирішень цього питання є використання мови PERL. Інтерпретатори мови PERL сьогодні існують для всіх операційних систем, а в таких системах як Linux та *BSD, UNIX PERL був і залишається невід'ємною частиною ОС [6–10].

Сьогодні мова PERL дозволяє реалізувати технічні задачі, для яких раніше потрібно було використовувати C/C++. Але на відміну від C/C++, програма мовою PERL не є файлом двійкового формату. Вона виконується за допомогою PERL – інтерпретатора, який, в свою чергу, підвантажує необхідні, специфічні для даної ОС, бібліотеки, що забезпечує високу переносимість коду та стабільність програми, не гіршу за стабільність самого PERL-інтерпретатора.

Для забезпечення багатозадачності вирішено використовувати принцип fork, тобто породження «клонів» процесу. На відміну від thread (поточного) принципу fork є зручнішим для вирішення серверних задач (приклади – Apache, Squid).

Ідеальним рішенням, звичайно, є використання як fork, так і thread, але використання thread, по-перше, не є оптимальним для розв'язання задачі створення PROXY-компоненти, по-друге, принцип thread сьогодні є відносно стабільно реалізованим лише в PERL під Linux. ActiveState (виробник PERL під Windows) оголосив про припинення розвитку thread-принципів. Реалізація thread в *BSD, UNIX залишається нестабільною не тільки в PERL-інтерпретаторі, але і у самій ОС.

Для розробки інтелектуального сервісу доступу до мережі Internet були обрані такі програмні засоби:

- Perl – для реалізації ядра системи і самого Proxy-сервера;
- PHP – для створення зручного користувацького інтерфейсу;
- MySQL – для збереження і обробки потрібних для системи даних.

Обрані програмні засоби належать до open source та freeware програмного забезпечення, завдяки чому набули великого поширення, кросплатформні та мають велику підтримку незалежних розробників. Інсталяція комплексу Perl + PHP + MySQL не є проблемою ні під ОС Windows, ні під ОС Linux, *BSD, UNIX. Програмний код інтелектуального сервісу працює під цими ОС практично без необхідності внесення змін.

Параметри інтелектуального сервісу:

- індивідуальні та загальні правила фільтрації та блокування ресурсів;
- правила анонімізації користувачів;
- корисувачі інтелектуального сервісу;
- правила маршрутизації та кешування HTTP-трафіку;
- об'єм використаного трафіка та квоти користувачів зберігаються у MySQL БД.

Адміністрування параметрів здійснюється користувачами та адміністратором інтелектуального сервісу через web-інтерфейс. Для реалізації адміністрування параметрів через web-інтерфейс використовується web-сервер, який підтримує PHP та броузер.

Реалізація багатозадачності

Необхідною вимогою до інтелектуального сервісу багатокористувацького доступу до Інтернет є можливість одночасного обслуговування багатьох користувачів. З цією метою використовується паралельне виконання запитів до інтелектуального сервісу [1, 2].

Багатозадачність компоненти реалізована за принципом fork() і побудована за такою схемою:

1. Запуск програми;
2. Створення процесу-нащадка за допомогою виклику fork();
3. Припинення роботи процесу-батька (процес-нащадок звільняє робочу консоль, підхоплюється головним процесом системи Init та продовжує роботу);
4. За допомогою PERL-класу socket() (аналог socket в C/C++, забезпечує роботу із TCP/IP протоколом), процес починає прослуховувати заданий порт та отримує запит клієнта.

Після отримання запиту мав би породжуватися процес-нащадок, що продовжував би працювати із під'єднаним клієнтом. Але після закінчення роботи нащадка процес-батько мав би викликати функцію wait() або waitpid() для запобігання перетворення нащадка на "зомбі". Цього можна було б досягти за допомогою обміну сигналами із батьківським процесом та процесом-нащадком. Але цей принцип є достатньо складним та досить нестабільним. Тому процес-нащадок породжує в свою чергу ще один процес (процес-онук для головного процесу) та закінчує роботу. За принципом fork() процес-нащадок, батько якого припинив роботу, "підхоплюється" головним системним процесом init та продовжує працювати. Тим самим забезпечується абсолютно незалежна робота процесу-батька (що приймає запити клієнтів) та процесів-онуків (що обробляють ці запити).

Опис функції fork()

fork() викликає створення нового процесу. Новий процес (процес-нащадок) є точною копією викликаючого процесу (процес-батько) за винятком таких параметрів:

- процес-нащадок має унікальний ID.
- процес-нащадок має інший батьківський процес ID (тобто ID процесу-батька).
- процес-нащадок має власну копію дескрипторів об'єктів батьківського процесу.
- процес-нащадок має нульове значення утилізації процесорного часу системи

Результати виконання

При успішному завершенні fork() повертає значення 0 до процесу-нащадка і ID процесу-нащадка до процесу-батька. Значення – 1, повернене до батьківського процесу, вказує, що процес-нащадок не створений, і глобальна змінна errno містить значення помилки.

Помилки, що можуть виникнути під час виконання:

- [EAGAIN] –за параметрами, вказаними в ядрі, система не може створити більше процесів;
- [EAGAIN] – користувач, програма якого викликає fork(), не є суперкористувачем (для UNIX систем – root), і виклик fork() перевищує загальне обмеження на кількість всіх користувацьких процесів, що виконуються одночасно;

• [EAGAIN] – користувач, програма якого викликає fork(), не є суперкористувачем (для UNIX систем – root) і виклик fork() перевищує його обмеження на кількість процесів, що виконуються одночасно;

• [ENOMEM] – недостатньо ресурсів пам'яті в системі.

Реалізація алгоритму аналізу правил

Алгоритм реалізований так:

функція get_remote згідно із заданими параметрами займається пошуком правил у базі даних, що відповідають заданим параметрам. Правила відбираються за допомогою SQL-запиту з умовою reversed like.

Приклад:

select * from rules where 'http:///www.ukr.net/banner.gif' like astr;

При знаходженні правил типу "Відключити поле" функція встановлює глобальну змінну цього поля у 0, після чого головна програма за допомогою регулярних виразів виду

/поле.*[\n\r]//g

замінює або відкидає задане поле із заголовка запиту.

При знаходженні правил "перенаправити на proxy" або "заблокувати", функція повертає значення адреси proxy-сервера або службове слово "banned".

Якщо правил не знайдено, повертається адреса proxy-сервера, що використовується за замовчуванням.

Висновки

Досліджено технології надання доступу до Інтернет, введено деякі правила користування глобальною мережею, які дозволять суттєво збільшити ефективність використання виробничих Інтернет-каналів. Головними технологіями, які використані при побудові інтелектуального сервісу багатокористувацького доступу до Інтернет, є: інтелектуальна фільтрація контента, інтелектуальна маршрутизація та облік трафіка, блокування доступу та анонімізація.

Фільтрація контента дозволяє здійснити (відповідно до політики безпеки) заборону доступу до певних ресурсів, розважальних веб-сайтів в робочий час тощо. Фільтрація дозволяє гнучко маніпулювати веб-контентом, вирізаючі рекламні банери та певні типи файлів, чим досягається економія трафіка.

Маршрутизація вихідного HTTP трафіка пришвидшує доступ користувачів до інформації Web-сайтів та зменшує затрати на зовнішній трафік.

Анонімізація доступу дозволяє уникнути передачі конфіденційної інформації рекламним агентствам, спамерам, хакерам та іншим зловмисникам у мережі. Прозорість запитів, що надходять через сервіс, дозволяє комп'ютерам локальної мережі отримати повноцінний доступ до ресурсів Інтернет.

1. Матеріали конференцій comp.unix.programming (news://news.demos.net/). 2. Матеріали конференцій fido7.ru.perl.guru (news://news.demos.net/). 3. Тарасов Д.О. Моделювання інформаційної інфраструктури комп'ютерних мереж та інформаційна безпека // Вісник Національного університету "Львівська політехніка". – 2002. – №464. – С. 302–311. 4. RFC 2617. HTTP Authentication: Basic and Digest Access Authentication, 1999. 5. RFC 2616. Hypertext Transfer Protocol – HTTP/1.1, 1999. 6. O'Relly Book. Network programming with PERL. 1998. 7. http://php.spb.ru/ 01.09.2003. 8. http://php.fud.ru/main/docum/10/. 9. http://php.fud.ru/main/docum/1/. 10. http://perl.far.ru /perl/docum/