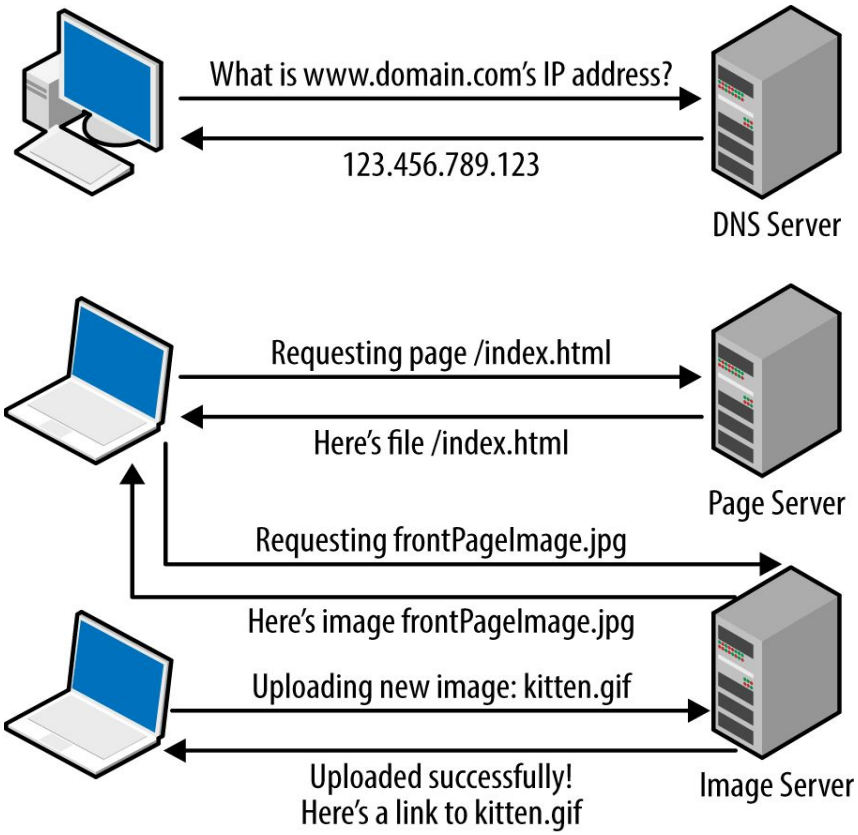
Web-scraping

Распространенные транзакции в сети



Известные прикладные протоколы и их стандартные порты:

- HTTP — основной протокол всемирной паутины (TCP порт 80)
- SMTP — протокол пересылки почты (TCP порт 25)
- FTP — протокол передачи файлов (TCP порт 21)

Структура HTTP-сообщения

- Стартовая строка (англ. Starting line) — определяет тип сообщения

Метод URI HTTP/Версия

Методы: GET, POST, HEAD, OPTIONS, PUT, DELETE

- Заголовки (англ. Headers) — характеризуют тело сообщения, параметры передачи и прочие сведения

- Тело сообщения (англ. Message Body) — непосредственно данные сообщения (например, данные с формы обратной связи или запрошенная html-страница)

Пример:

```
GET /wiki/HTTP HTTP/1.0
Host: ru.wikipedia.org
```

HTTP-ответ

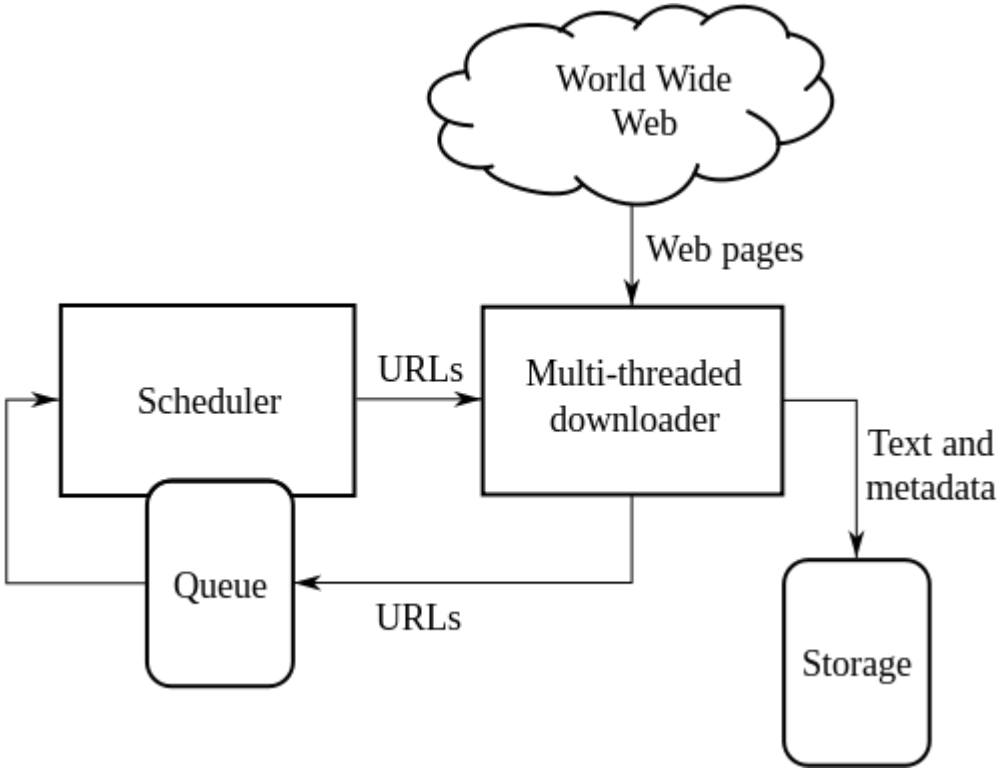
- Версия — пара разделённых точкой цифр как в запросе
- Код состояния (Status Code) — три цифры

1xx (инфомационные)
2xx (успех)
3xx (redirect)
4xx (ошибка клиента)
5xx (ошибка сервера)

- Пояснение (Reason Phrase) — текстовое короткое пояснение к коду ответа для пользователя.

Примеры: `HTTP/1.0 200 OK`, `403 Access allowed only for registered users`

Типичный веб-краулер



Стадии скрапинга

- Получение html-страницы.
- Парсинг страницы
- Сохранение полученных данных.
- (Опционально) Переход на другую страницу и повтор процесса.

Получение страницы (wget/curl, общее и различия)

- Обе утилиты могут скачивать контент через FTP, HTTP and HTTPS
- Обе утилиты могут выполнять POST-запросы
- Обе утилиты поддерживают cookies
- Обе open software

wget:

- только в командной строке
- рекурсивность (!!!)
- поддерживает только базовую HTTP авторизацию
- может быть набран только левой рукой %)

Получение страницы (wget/curl, общее и различия)

curl:

- использует библиотеку libcurl (кроссплатформенная, есть api)
- работает в более традиционном unix-like стиле (через pipe)
- больше адаптирована для разового скачивание без рекурсии
- поддерживает больше протоколов (FTP, FTPS, Gopher, HTTP, HTTPS, SCP, SFTP, TFTP, TELNET, DICT, LDAP, LDAPS, FILE, POP3, IMAP, SMB/CIFS, SMTP, RTMP и RTSP)
- присутствует на большем количестве ОС (OS/400)
- поддерживает SOCKS (прокси)
- поддерживает больше протоколов аутентификации (NTLM) и SSL-библиотек


CURL
WGET
MARVEL
CAPTAIN AMERICA
CIVIL WAR
MAY 6, 2016
imgflip.com

Получение страницы (языки)

Python: PycURL (libcurl) / встроенный urllib

C/C++: cURL(libcurl) / более экзотичные варианты (boost / poco)

```
#include <iostream>
#include <string>
#include <boost/asio.hpp>
#include <boost/regex.hpp>
int main()
{
    boost::asio::ip::tcp::iostream s("tycho.usno.navy.mil', "http");
    if(!s)
        std::cout << "Could not connect to tycho.usno.navy.mil\n;
    s  << "GET /cgi-bin/timer.pl HTTP/1.0\r\n"
       << "Host: tycho.usno.navy.mil\r\n"
       << "Accept: */*\r\n"
       << "Connection: close\r\n\r\n' ;
...
```

Парсинг страницы (python)

- Регулярные выражения (re.compile, re.match, re.search, re.split)
- Автоматический HTML/XML парсер (Beautiful soup)
- Библиотеки обработки XML (встроенная xml.etree.ElementTree или lxml)

HTTP cookie

Фрагмент данных, сохраняемый сервером на клиентской стороне.

Применение:

- аутентификация
- хранения персональных предпочтений и настроек пользователя
- отслеживание сеанса доступа
- ведение статистики

Упрощения

- Ищите мобильную версию или версию для печати
- Смотрите, что в js
- В URL может быть полезная информация
- Пробуйте отыскать другие ресурсы с нужной информацией

Использование API

API - это **определенный** набор HTTP-запросов, а также определение структуры HTTP-ответов.

- Форматы ответов: json, xml (json занимает меньше места при сохранении вложенности)
- Зачастую необходима авторизация (через ключ-токен)

```
token = "<your api key>"
webRequest = urllib.request.Request("http://myapi.com", headers={"token":token})
html = urlopen(webRequest)
```

Работа с файлами

- Документы (редакторы + встроенные функции encode/decode)
- csv (встроенная библиотека)
- pdf (pdfminer3k)
- docx (import zipfile + дальнейший парсинг xml)

Страницы с javascript. Динамические страницы.

Проблема: рано или поздно столкнемся с тем, что получаем не то, что видим через браузер, так как скрапер не выполнил js-скрипт

Решение:

- Скрапить js напрямую
- Каким-то образом выполнить скрипт: selenium (автоматизация работы с браузером) + headless browser (phantomjs)
- Для ajax возможен реверс-инжиниринг страницы через консоль браузера и обнаружение запросов вида `http://example.webscraping.com/ajax/search.json?page=0&page_size=10&search_term=a`

Продвинутый скрапинг. Распознавание капч.

- Tesseract (в форме утилиты командной строки или в форме библиотеки pytesseract)
- Сервисы рещения капч (https://www.9kw.eu/)

Дополнительные советы

- Устанавливайте human-like http-заголовки
- Смотрите и используйте куки
- Ищите и избегайте honeypot (скрытые от браузера линки, переход по которым вызывает блокировку)