HTTP protokol, RESTful, WebSocket

Mobilné technológie a aplikácie

FIIT STU v Bratislave

prof. Ing. Ivan Kotuliak, PhD.

Ing. Marek Galinski

HTTP protokol

HTTP – Základné informácie

- HTTP – HyperText Transfer Protocol
  - Štandardizovaný v roku 1997 dokumentom RFC 2068
    - 1999 RFC 2616, 2014 RFC 7230
  - OSI L7 protokol (aplikačná vrstva)
  - Štandardne využívajúci TCP/80

HTTP – Základné informácie

- Dopytovo orientovaný (request-response protocol)
  - HTTP Request
  - HTTP Response
- Plaintext protokol
  - Implicitne nešifrovaný
  - Sám o sebe šifrovať nevie (treba sa spoľahnúť ina iné protokoly)

HTTP – Verzie protokolu

- HTTP/1.0 (pôvodná verzia)
- HTTP/1.1 (štandard)
- HTTP/2 (2015)
- HTTP/3 (QUIC) (2018)

HTTP – Request

- Method
  - GET, POST, PUT, DELETE, XXXXXXX
- URL/URI
  - IP / FQDN : [PORT] : [PATH]
- Header
  - Authorization, Application-Type, Timeout, etc ...
- Body
  - ....

HTTP – Request metódy

- GET – získanie dát zo servera, spravidla neobsahuje nič v body
- POST – uloženie dát na server, musí obsahovať body
- PUT – update scenár, aktualizácia dát na serveri
- DELETE – odstránenie dát zo servera
- PATCH – čiastková úprava entity
- CONNECT – konvertuje request na vytvorenie TCP/IP tunela

HTTP – Request metódy

- OPTION – Zistenie povolených metód pre dopytovanú URL
- TRACE – Echo HTTP Requestu (kontrola komunikácie, debug)
  - Kontrola či po ceste nenastali nejaké zmeny
  - Z bezpečnostných dôvodov by malo byť použitie na serveri zakázané
  - Prezrádza viac, než by musel (uložené cookies, cache, ...)

HTTP – Request metódy

STU FIIT

HTTP method	RFC	Request has Body	Response has Body
GET	RFC 7231	No	Yes
HEAD	RFC 7231	No	No
POST	RFC 7231	Yes	Yes
PUT	RFC 7231	Yes	Yes
DELETE	RFC 7231	No	Yes
CONNECT	RFC 7231	No	Yes
OPTIONS	RFC 7231	Optional	Yes
TRACE	RFC 7231	No	Yes
PATCH	RFC 5789	Yes	Yes

HTTP – Request metódy

- Mnoho služieb ktoré poskytujú HTTP API využívajú na všetky typy správ metódu POST
  - Application-Type: text/json
  - V samotnom JSONe v body požiadavky sa nachádza definícia metódy
    - {method: insert}, {method: get}, ...

HTTP – Request polia

- Accept-Encoding: gzip, deflate
- Content-Length: 348
- Content-MD5: Q2hlY2sgSW50ZWdyaXR5IQ
- Content-Type: application/x-www-form-urlencoded
- Referer: http://en.wikipedia.org/wiki/Main_Page
- User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/12.0

HTTP – Response

- Code
  - 1xx, 2xx, 3xx, 4xx, 5xx
- Header
  - Authorization, Application-Type, Timeout, etc ...
- Body
  - ….

HTTP – Ukážka request & response

- Request

```
GET /index.html HTTP/1.1
Host: www.example.com
```

HTTP – Ukážka request & response

- Response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 138
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html>
<head>
  <title>An Example Page</title>
</head>
<body>
  Hello World, this is a very simple HTML document.
</body>
</html>
```

HTTP – Response codes

- 1xx – informačné / dočasné správy
- 2xx – potvrdzovacie správy
- 3xx - presmerovania
- 4xx – chybové stavy (klientské, spravidla)
- 5xx – chybové stavy (serverové)

HTTP – Response codes – 1xx

- Dočasné správy slúžiace na informáciu o tom, že sa niečo deje, ale ešte sa to neudialo do konca
  - 100 Continue
    - Kontrola hlavičiek „Expect: 100-continue“
  - 101 Switching Protocols
  - 102 Processing
    - Prevencia proti timeoutu

HTTP – Response codes – 2xx

- Potvrdzovacie správy spravidla znamenajúce že požiadavka bola vybavená korektne
  - 200 OK
  - 201 Created
  - 202 Accepted
  - 204 No-content

204

No Content

HTTP – Response codes – 3xx

- Správy informujúce klienta o presmerovaní alebo zmene
  - 300 Multiple Choices
  - 301 Moved Permanently
  - 302 Found
  - 304 Not Modified
  - 307 Temporary Redirect
  - 308 Permanent Redirect

301

Moved Permanently

HTTP – Response codes – 4xx

- Chybové správy informujúce, že server požiadavku prijal, ale z nejakého dôvodu ju nemôže obslúžiť
  - 400 Bad Request
  - 401 Unauthorized
  - 402 Payment required
  - 403 Forbidden
  - 404 Not Found

HTTP – Response codes – 4xx

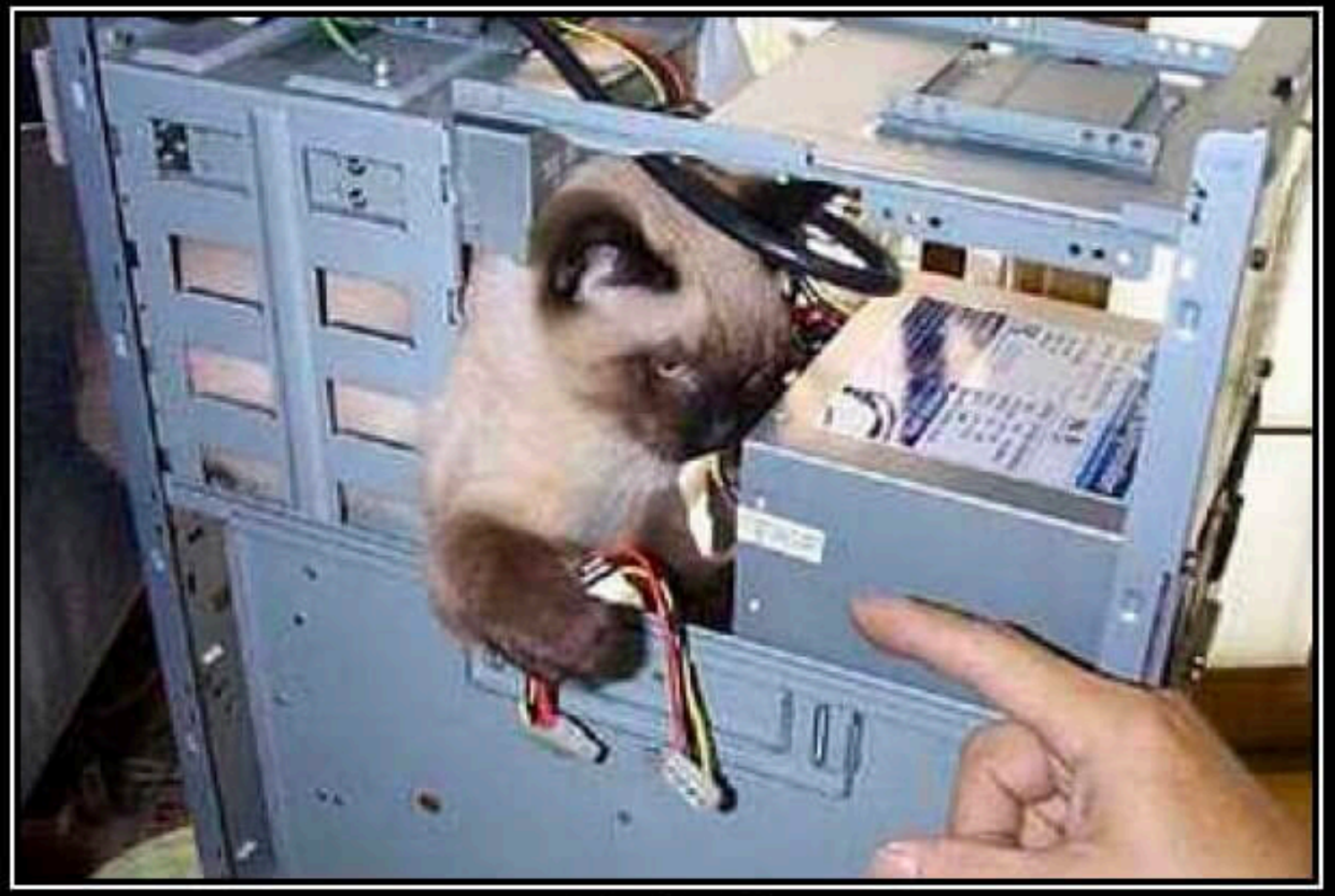
- 405 Method Not Allowed
- 408 Request Timeout
- 409 Conflict
- 415 Unsupported Media Type
- 426 Upgrade Required

415

Unsupported Media Type

HTTP – Response codes – 5xx

- Chybové správy informujúce, že server požiadavku nemôže spracovať, pretože niečo na serveri nefunguje správne
  - 500 Internal Server Error
  - 501 Not implemented
  - 502 Bad Gateway
  - 503 Service Unavailable



500

Internal Server Error

HTTP – Bizzare codes

- 429 Too Many Requests
  - V prípade služieb, kde používateľ má definované určité limity
- 451 Unavailable for Legal Reasons
  - Poskytovateľ odmietne spracovať požiadavku, pretože na to má zákonný dôvod
  - Ray Bradbury - Fahrenheit 451 (cenzúra, pálenie kníh)

HTTP – Bizzare codes

- 418 I‘m a teapot
  - “The RFC specifies this code should be returned by teapots requested to brew coffee“
  - RFC 2324 (vydané 1 apríla 1998)

418

I'm a teapot

HTTP – Unofficial codes

- 526 Invalid Certificate (Cloudflare)
- 494 Request Header Too Long (nginx)
- 495 SSL Certificate Error (nginx)
- 450 Blocked by Windows Parental Controls (Microsoft)

HTTP – Rozšírenia

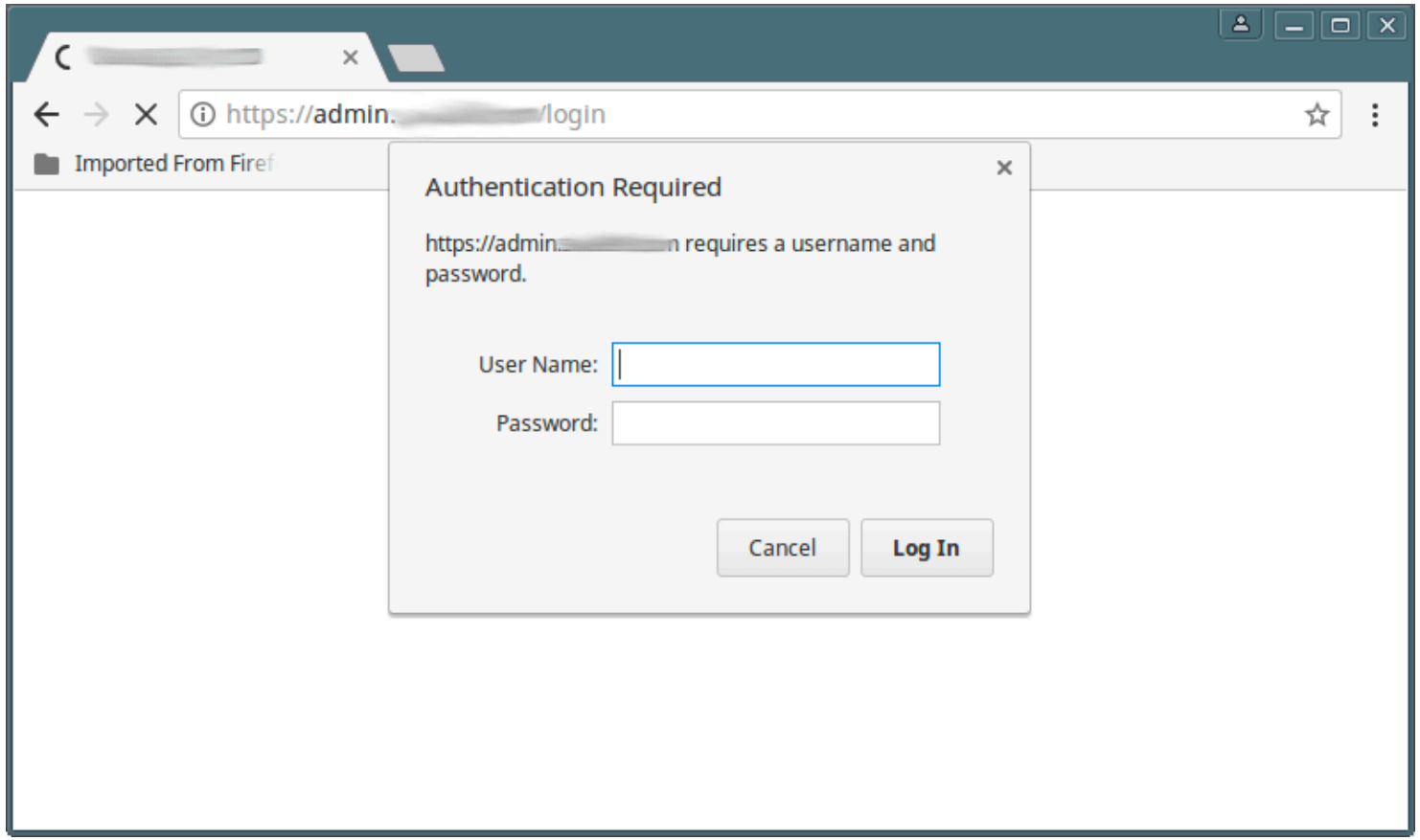
- HTTPS – HTTP protokol šifrovaný pomocou SSL/TLS
  - Definovaný v RFC 2818
  - Štandardne využívajúci TCP/443
  - Bezpečnosť vyhodnocovaná certifikátmi

HTTP – Autentifikácia

- HTTP sám o sebe podporuje dva spôsoby autentifikácie
  - Basic access authentication
  - Digest access authentication

HTTP – Basic access authentication

- Najjednoduchší spôsob autentifikácie
  - Nevyžaduje cookies, login stránky, session, ...



HTTP – Basic access authentication

- **Nie je šifrovaný (Base64)**
- Bez HTTPS bezpečnosť nulová
- Neexistuje logout metóda
  - Sú rôzne workaroundy
    - Opakovaný request so zlým heslom
    - Clear cache

HTTP – Digest access authentication

- Využíva MD5 hash
  - Bezpečnosť otázna
- Hash počítaný z výzvy, ktorú poslal server a mena a hesla
- Rozšírená bezpečnosť
  - Prevencia proti Rainbow Tables
  - Timestamp – prevencia proti opakovaným útokom

HTTP – Digest access authentication

- Nie je odolný voči Man-in-the-middle útokom
- Nepodporuje silnejšie kryptografické mechanizmy
  - bcrypt, SHA-1, ...

HTTP – Digest access authentication

- Ukážka

```
GET /dir/index.html HTTP/1.0
Host: localhost
```

```
HTTP/1.0 401 Unauthorized
Server: HTTPd/0.9
Date: Sun, 10 Apr 2014 20:26:47 GMT
WWW-Authenticate: Digest realm="testrealm@host.com",
                        qop="auth,auth-int",
                        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
                        opaque="5ccc069c403ebaf9f0171e9517f40e41"
Content-Type: text/html
Content-Length: 153

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Error</title>
  </head>
  <body>
    <h1>401 Unauthorized.</h1>
```

HTTP – Digest access authentication

- Ukážka

```
GET /dir/index.html HTTP/1.0
Host: localhost
Authorization: Digest username="Mufasa",
                      realm="testrealm@host.com",
                      nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
                      uri="/dir/index.html",
                      qop=auth,
                      nc=00000001,
                      cnonce="0a4f113b",
                      response="6629fae49393a05397450978507c4ef1",
                      opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

```
HTTP/1.0 200 OK
Server: HTTPd/0.9
Date: Sun, 10 Apr 2005 20:27:03 GMT
Content-Type: text/html
Content-Length: 7984
```

HTTP – Kompresia

- HTTP podporuje kompresiu správ, ak sa na tom dohodnú obe strany
- Optimalizácia rýchlosti
- Optimalizácia množstva prenášaných dát
- Podporované všetkými hlavnými webovými servermi

HTTP – Kompresia

- Ukážka

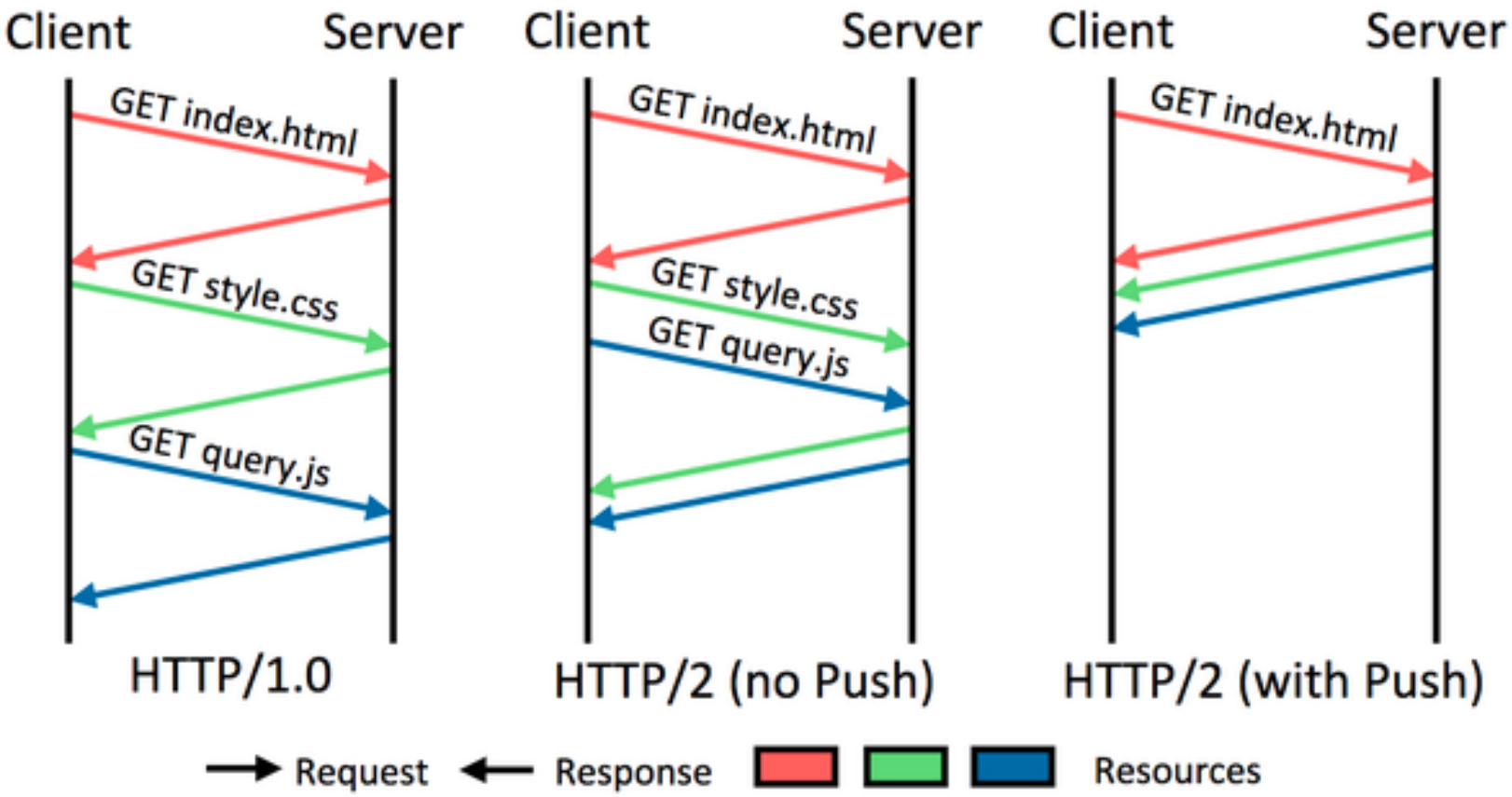
```
GET /encrypted-area HTTP/1.1
Host: www.example.com
Accept-Encoding: gzip, deflate
```

```
HTTP/1.1 200 OK
Date: mon, 26 June 2016 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix)  (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
```

HTTP/2

- Rozdiely oproti HTTP/1.1
  - Minifikácia prenášaného kódu
  - Push – server vie vrátit klientovi viac informácií, než si vyžiadal (bez requestu)
  - Kompresia HTTP hlavičiek
  - HTTPS *de facto* povinné
    - Štandaridizované pre HTTP aj HTTPS, vačšina implementácií však existuje iba pre HTTPS

HTTP/1.1 vs HTTP/2 + Push



HTTP/3 (QUIC)

- Pôvodne boli HTTP/3 a QUIC dva nezávislé projekty
- QUIC – interný projekt Google (2012)
- November 2018 – „HTTP/3 bude QUIC“
- **Postavený nad UDP**
  - Paralelné spojenia na jednotlivé dátové entity

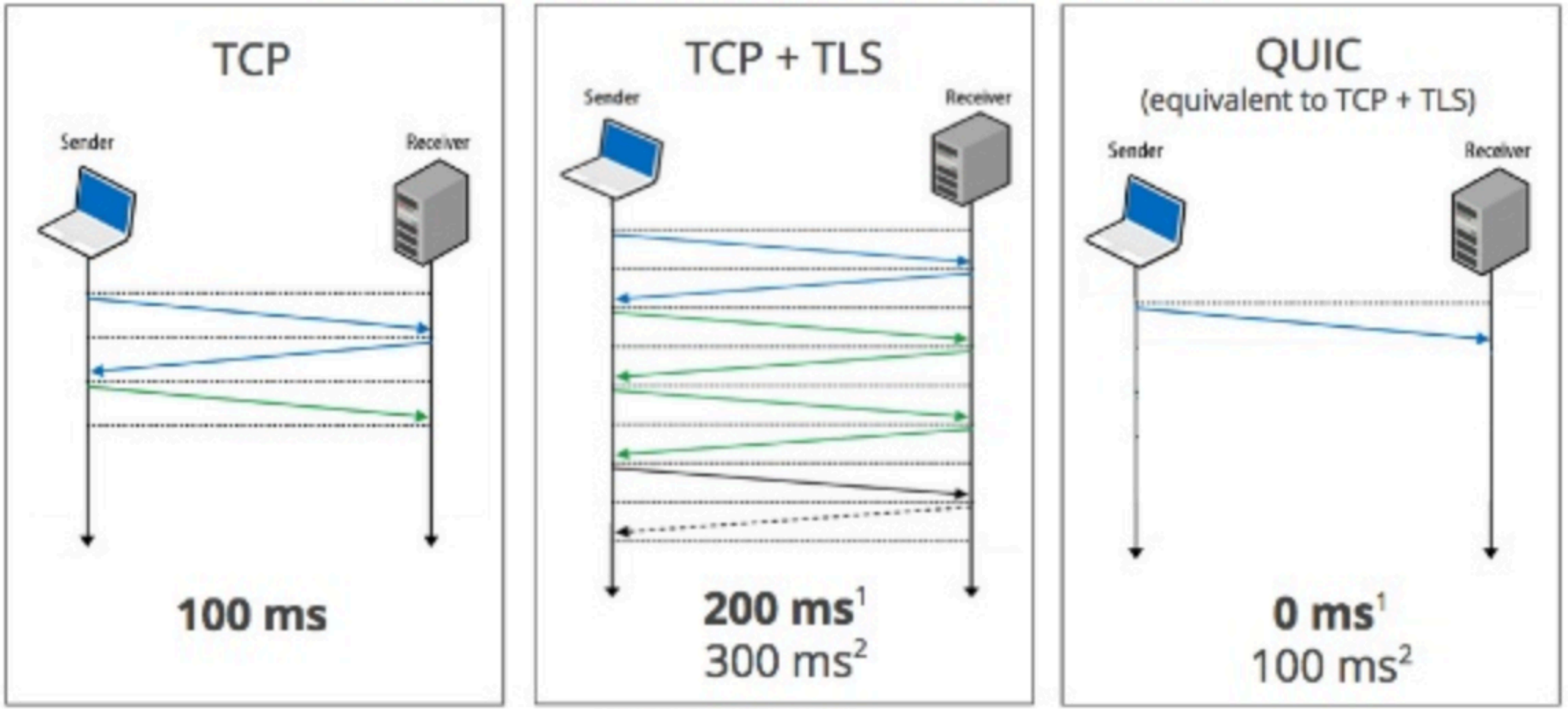
HTTP/3 (QUIC)

- Redukuje počet nastavovacích a kontrolných správ
  - Nadviazanie šifrovaného spojenia so všetkými parametrami jednou úvodnou správou
  - Kontrola chýb riešená priamo QUIC protokolom, nie spoliehajúca sa na nižší protokol

HTTP/3 (QUIC)

- Redukuje počet nastavovacích a kontrolných správ
  - Pokročilá detekcia chýb (FEC – Forward error correction)
    - Samoopravné kódy – redundatný spôsob, každá strana vie opraviť určité množstvo chýb bez retransmisie

HTTP vs QUIC


TCP
Sender
Receiver
100 ms
TCP + TLS
Sender
Receiver
200 ms[1]
300 ms[2]
QUIC
(equivalent to TCP + TLS)
Sender
Receiver
0 ms[1]
100 ms[2]

HTTP/3 (QUIC)

- Podpora
  - Chromium 29 - default
  - Google Chrome – dá sa zapnúť
  - Opera 16 – dá sa zapnúť
  - Firefox – nie
  - Apache, Nginx - nie

REST

REST - úvod

- REST – Representation State Transfer
  - Architektonický štýl určený na poskytovanie webových služieb
  - Alternatíva ku SOAP
  - Predstavený v roku 2000
    - Počiatky v roku 1994, dizertačná práca

REST - ciele

- Objektový model nad HTTP
- Rozširovanie alebo zmena sady funkcionality aj za behu aplikácie
- Uniformné rozhranie
- Portabilita, spoľahlivosť pri zlyhaní časti systému

REST - princípy

- Klient – Server model
  - Výmena údajov na princípe požiadavka, odpoveď
  - Oddelenie klientskej logiky od serverovej logiky
  - Multiplatformovosť

REST - princípy

- Statelessness (Bezstavovosť)
  - Jednotlivé komunikácie nemajú stavy a nie sú od seba závislé
  - Každý request obsahuje dostatok dát na to aby mohol byť obslúžený (dostať response)

REST - princípy

- Cache (krátkodobá pamäť)
  - Dá sa zadefinovať, ktorý obsah chceme cacheovať a následne obsluhovať efektívnejšie
    - Obrázky, ktoré sa nemenia každú chvíľu
    - Trvalý obsah

REST - princípy

- Vrstvovitosť (Layered system)
  - Klient nevie, s kým komunikuje
    - Proxy? Konečný server? Medzičlánok?
  - Škálovateľnosť
  - Bezpečnosť
    - Nepristupuje sa priamo k databáze
    - Security policy (read-only, autorizácia, etc ...)

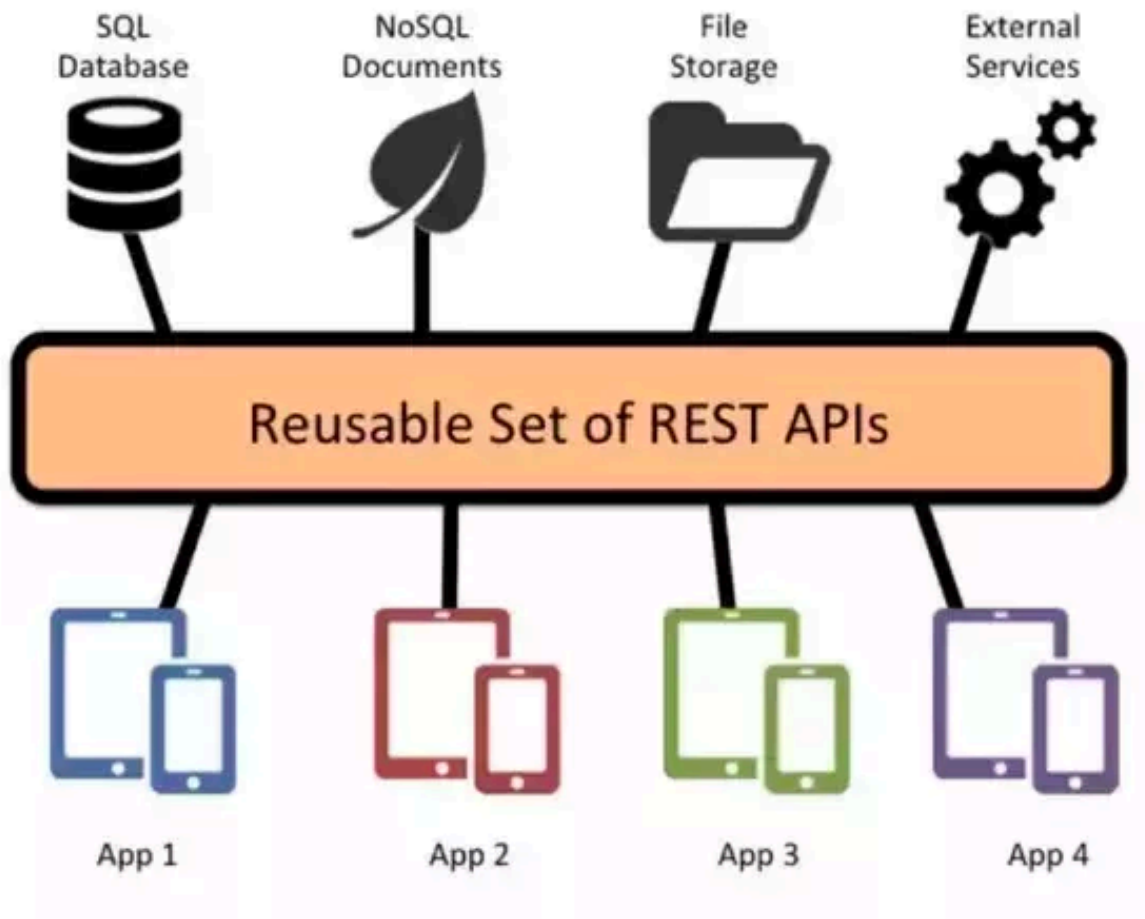
REST - princípy

- Code-on-demand
  - Pomocou RESTu sa dá posielať kód vykonávateľný na klientovi
    - napr. JavaScript

REST - princípy

- Uniformné rozhranie
  - Každá časť väčšieho systému sa môže vyvíjať samostatne
  - Jednotlivé časti nie sú na seba viazané technológiou

REST - koncept



WebSocket

WebSocket - úvod

- WebSocket – komunikačný protokol podporujúci full-duplex nad TCP
- Štandardizovaný v RFC 6455 v roku 2011

- **WebSocket nie je HTTP**
- **WebSocket nie je nad HTTP**

WebSocket - úvod

- Využíva rovnaké porty ako HTTP a HTTPS
  - 80 HTTP, 443 HTTPS
- WebSocket Handshake využíva HTTP Upgrade správu
  - Zaručenie kompatibilty
- Podporovaný všetkými hlavnými prehliadačmi
  - Dokonca aj MS Edge a MS Internet Explorer

WebSocket - úvod

- Na rozdiel od HTTP a HTTPS komunikácia je obojstranná bez dvojíc request-response
- ws:// a wss://

WebSocket – nadviazanie spojenia


```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

WebSocket – ukážka kódu - posielanie


```
// Send a message when the form is submitted.
form.onsubmit = function(e) {
  e.preventDefault();

  // Retrieve the message from the textarea.
  var message = messageField.value;

  // Send the message through the WebSocket.
  socket.send(message);

  // Add the message to the messages list.
  messagesList.innerHTML += '<li class="sent"><span>Sent:</span>' + message +
                            '</li>';

  // Clear out the message field.
  messageField.value = '';

  return false;
};
```

```
socket.send(message);
```

WebSocket – ukážka kódu - prijímanie


```
socket.onmessage = function(event) {
```

```
// Handle messages sent by the server.
socket.onmessage = function(event) {
  var message = event.data;
  messagesList.innerHTML += '<li class="received"><span>Received:</span>' +
                            message + '</li>';
};
```

WebSocket - knižnice

- C++ - libwebsockets
- Java – jetty
- NodeJS – ws
- Ruby – em-websocket
- Python – pywebsocket
- PHP - phpws