Ministerstvo financií
Slovenskej republiky

EFQM
Recognised for excellence
5 star - 2012

Siete, Internet a telekomunikácie
Protokol HTTP

Ladislav Hudec

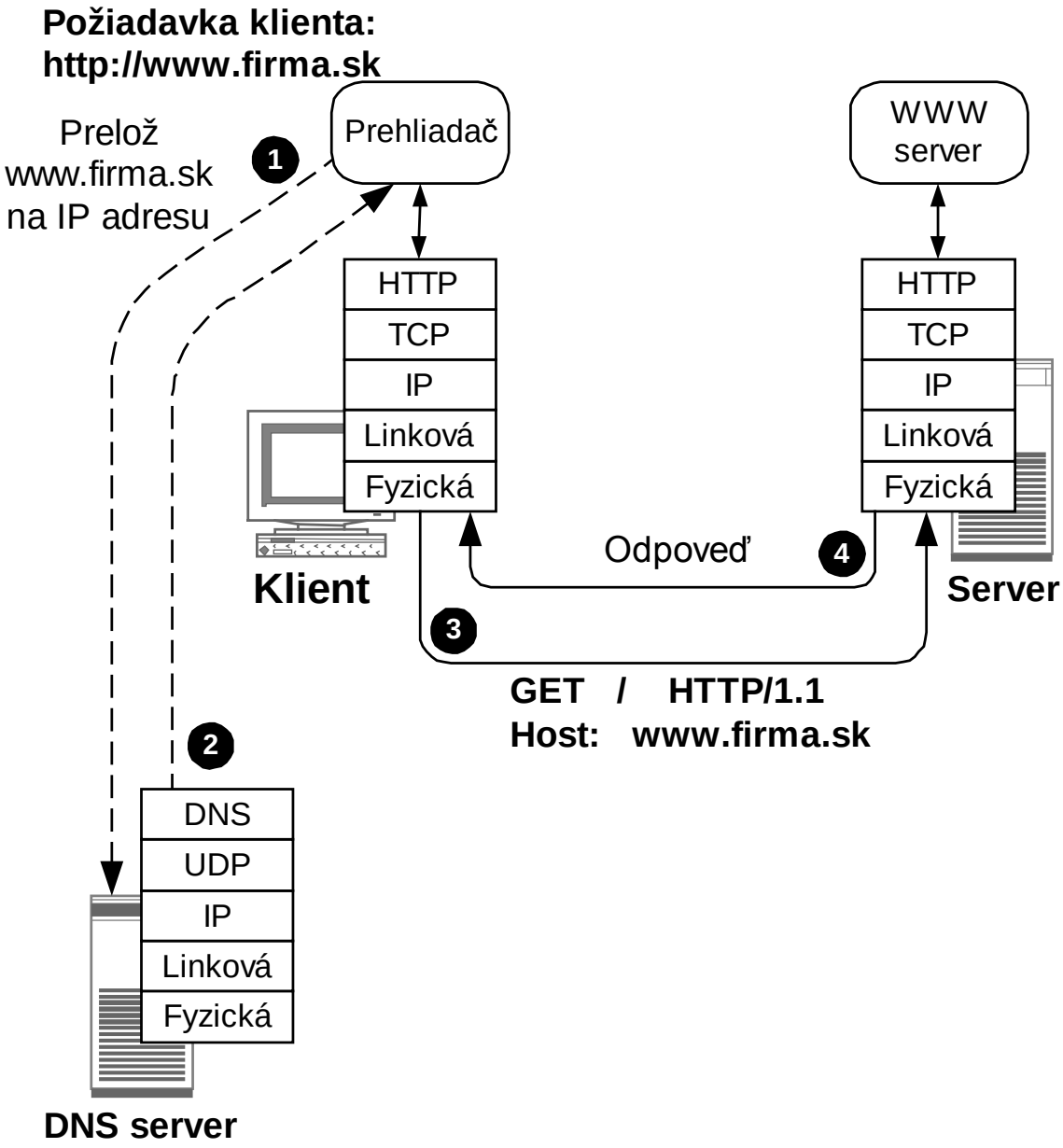
2013

KPMG
cutting through complexity™

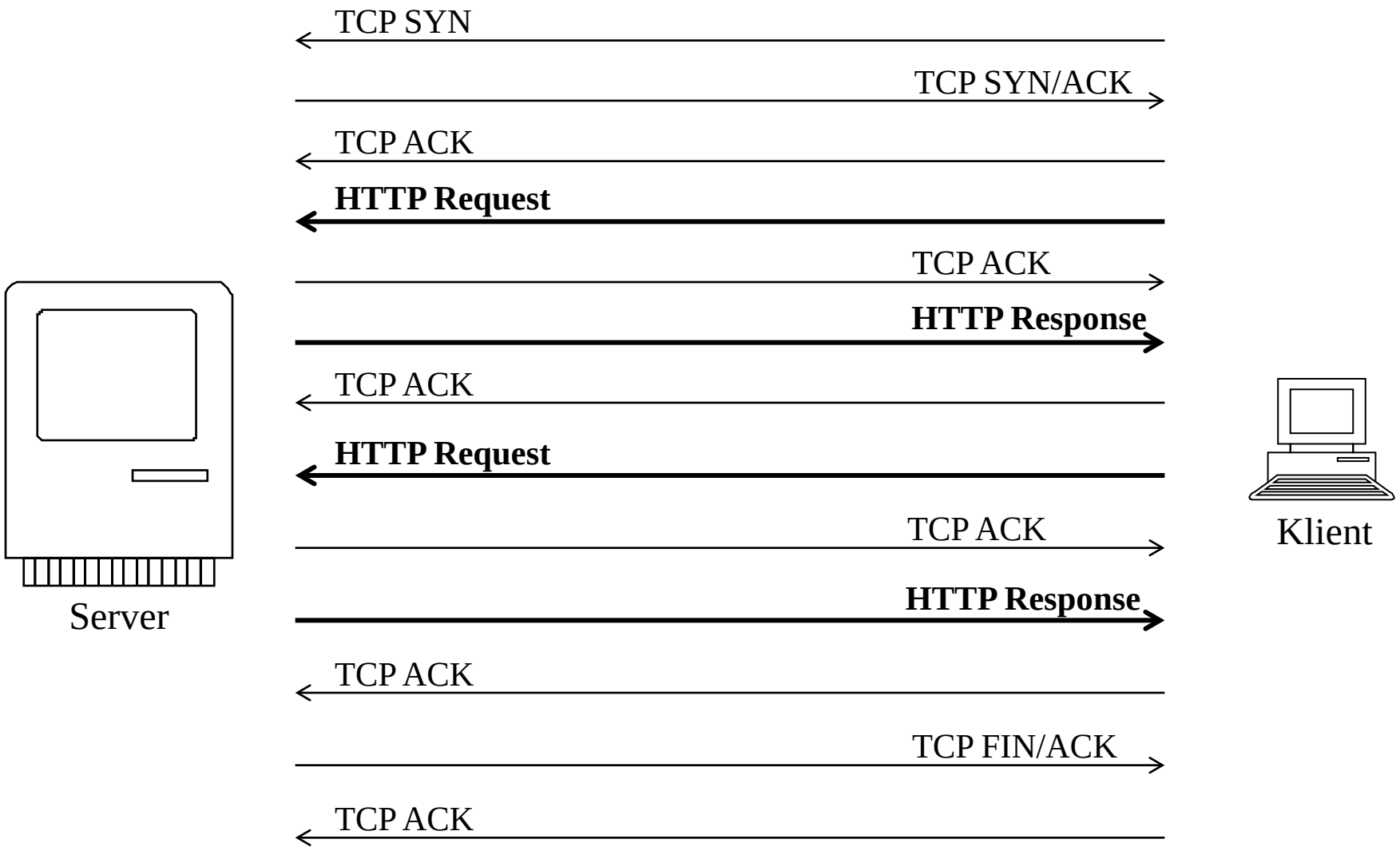
Protokol HTTP

- Protokol HTTP (*Hypertext Transfer Protocol*) bol navrhnutý v roku 1990. Ďalším medzníkom bol protokol HTTP verzia 0.9, ktorý bol veľmi jednoduchý, preto zaznamenal veľké množstvo implementácií.
- Koncepcia Klient/server
  - Základnou architektúrou komunikácie v protokole HTTP je komunikácia **klient / server**. V prípade, že sa nadväzuje priame spojenie protokolom TCP medzi klientom a serverom (viď nasledujúci obrázok), potom používateľ zapíše do okna prehliadača identifikátor objektu (URI), ktorého chce prehliadať, a klient najprv z identifikátora objektu vyberie meno servera a preloží ho pomocou DNS na IP adresu (1 a 2). Potom klient nadviaže s takto získanou IP adresou servera spojenie protokolom TCP.
  - Do takto vytvoreného kanála vloží prehliadač HTTP dopyt (3), na ktorý v tom istom spojení server odpovie HTTP odpoveďou (4). Prehliadač následne zobrazí odpoveď používateľovi. Dôležité je, že prehliadač zobrazuje používateľovi webové stránky. Každá webová stránka sa väčšinou skladá z radu objektov. Každý objekt je nevyhnutné z webového servera stiahnúť jedným HTTP dopytom. V protokole HTTP starších verzií sa pre každý dopyt vždy nadväzovalo nové TCP spojenie. Takže prvým dopytom sa stiahol základný text stránky, v ktorom bol rad ďalších odkazov, ktoré bolo nevyhnutné stiahnuť pre zobrazenie stránky. V ďalšom kroku sa, pokiaľ to bolo možné, naraz nadviazalo s webovým serverom na stiahnutie každého objektu TCP spojenie. Takáto stratégia vedie k vytvoreniu špičky v záťaži prenosovej cesty.
  - Protokol HTTP verzie 1.1 implicitne predpokladá, že TCP spojenie bude medzi klientom a serverom nadviazané **jedno pre celú sadu dopytov** („pre celú webovú stránku“). Je možné ho uzavrieť i po jednom dopyte i po viacero dopytoch. Klient môže odoslať v jednom spojení viacero dopytov bez toho, aby vždy čakal na vybavenie predchádzajúci dopyt (*Pipelining*).

Protokol HTTP – architektúra protokolu



Protokol HTTP/1.1 – komunikácia klient/server



Protokol HTTP

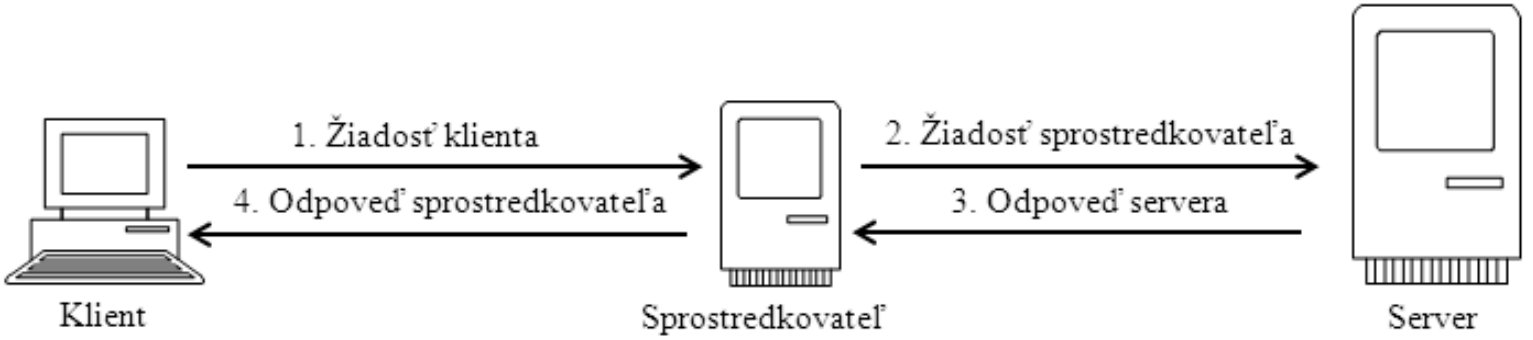
- Protokol HTTP verzia 1.1 implicitne predpokládá, že do vytvoreného spojenia sa vkladá viacero dopytov a odpovedí na ne. Protokol HTTP verzia 1.1 funguje štandardne s trvalými spojeniami. Pokiaľ je požadované spojenie treba explicitne ukončiť, potom treba do hlavičky vložiť príkaz **Connection: Close.**
- **Komunikácia v protokole HTTP sa zásadne skladá z dopytu a odpovedi.** Relácia medzi klientom a serverom je tvorená vždy iba dopytom a odpoveďou na tento dopyt. Staršie verzie protokolu HTTP dokonce nadväzovali spojenie protokolom TCP iba na jednu reláciu dopyt – odpoveď. Novšie verzie umožňjú využiť nadviazané spojenie na viacero relácií dopyt – odpoveď. Avšak i v prípade, že jedným TCP spojením prechádza viacero relácií, tak tieto relácie spolu nijako nesúvisia.
- Skutočnosť, že protokol HTTP neumožňuje dlhší dialóg než jeden dopyt a bezprostrednú odpoveď na neho, je istým obmedzujúcím limitom protokolu HTTP. Prakticky môže nastať nasledujúca situácia: používateľ chce pomocou protokolu HTTP nakupovať vo virtuálnom obchodnom dome na Internete. Vyberie si tovar, ktorý si uloží do virtuálneho nákupného košíka, ktery si nesie behom svojho nákupu. Lenže zákazník bude ďalšou reláciou vyberať ďalší tovar a ako si uchovať o anonymnom zákazníkovi informáciu, že má už nejaký tovar v košíku (kde ten košík udržovať?). Tento problém je riešený pomocou **Cookie**.
- Cookie je **uložený na klientskom počítači** a obsahuje dôležité informácie relevantné konkrétnej webovej aplikácii ako je meno zákazníka, položky v nákupnom košíku alebo meno a heslo. Údaje z cookie sú posielané späť na server s každou následnou žiadosťou s tým, že sa serveru dovoľuje tieto informácie aktualizovať a opäť poslať klientovi späť. Cookies takto umožňujú serverom pamätať si používateľské údaje medzi žiadosťami.

Protokol HTTP

- Iným obmedzením protokolu HTTP je použitie architektúry klient/server. Tá neumožňuje odosielať asynchrónne udalosti zo servera klientovi. Protokolom HTTP sa tak ťažko vytvárajú aplikácie typu „burza“. Tj. kde by pre používateľa bolo praktické, v prípade zaujímavej výmeny cenného papiera, aby server upozornil klienta na túto skutočnosť. Server môže túto skutočnosť tak oznámiť klientovi nejskôr v okamžiku, kedy klient odošle nejaký dopyt na server.
- Používateľ si väčšinou nastaví prehliadač (klienta) tak, že získanú požiadavku nielen používateľovi zobrazí, ale pokiaľ je to možné, tak ju tiež uloží na disk do pamäti cache. Pri opakovaní dopytu potom môže byť používateľovi informácia zobrazená priamo z lokálneho disku. Existujú nejrôznejšie stratégie, kedy zobrazovať tieto informácie a kedy má klient preniesť informácie zo servera. Dokonce existuje i možnosť, že sa **klient dopytuje servera**, či sa informácie nezmenili tým, že **zo servera prenesie iba hlavičku odpovedi**. Niektoré odpovede servera môžu byť označené ako neuložiteľné do pamäti cache, potom ich tam klient uložiť nesmie.
- Internetový prehliadač nebýva klientem iba protokolu HTTP. Spravidla sa jedná o integrovaného klienta, ktorý „vie“ i protokol FTP a prípadne vie vyvolať klientov ďalších protokolov:
  - Protokolov elektronickej pošty
  - Protokolu LDAP (tj. adresár)
  - Protokolu NNTP (tj. diskuzné skupiny)
  - Telnet apod.

Medziľahlé zariadenia - sprostredkovatelia

- Jednoduchá dvojica žiadosť HTTP / odpoveď HTTP medzi klientom a serverom sa stáva zložitejšou, keď sú vo virtuálnej komunikačnej ceste medzi klientom a serverom umiestnení **sprostredkovatelia** (intermediary). Ide o také zariadenia ako **proxy**, **brány** alebo **tunely**, ktoré sa používajú na zvýšenie výkonu (priepustnosti), zaisteniu bezpečnosti alebo vykonávajú iné potrebné funkcie pre konkrétnych klientov alebo servery. Proxy servery sa bežne používajú na webe, pretože môžu výrazne zlepšiť čas odozvy pre skupinu podobných klientskych počítačov.
- Keď je v komunikácii medzi klientom a serverom HTTP zapojený sprostredkovateľ HTTP komunikácie, tak klient komunikuje so serverom prostredníctvom sprostredkovateľa. To znamená, že všetka premávka (traffic) medzi klientom a serverom prechádza sprostredkovateľom. Táto skutočnosť umožňuje sprostredkovateľovi vykonávať **rôzne funkcie nad prechádzajúcou premávkou**, napríklad ukladanie do vyrovnávacej pamäti (cache) sprostredkovateľa, preklad, agregácie alebo zapúzdrenie. Na nasledujúcom obrázku je demonštrovaný príklad komunikácie medzi klientom a serverom HTTP prostredníctvom jedného sprostredkovateľa.



Medziľahlé zariadenia - sprostredkovatelia

- Jednoduchá dvojica žiadosť HTTP / odpoveď HTTP sa zmení na dve dvojice (štyri kroky):
  - Klient HTTP odošle správu žiadosti sprostredkujúcemu zariadeniu.
  - Sprostredkovateľ žiadosť spracuje, vykoná v prípade potreby zmeny v žiadosti, a potom pošle žiadosť na server.
  - Server HTTP prečíta a interpretuje žiadosť, vykoná príslušné akcie a odošle odpoveď. Pretože dostal svoju žiadosť od sprostredkovateľa, jeho odpoveď ide späť sprostredkovateľovi.
  - Sprostredkovateľ odpoveď spracuje, opäť prípadne urobí zmenu, a potom ju pošle klientovi.
- Uvedený príklad možno zovšeobecniť na **viacero sprostredkovateľov pri komunikácii medzi klientom a serverom**.
- Pri komunikácii medzi klientom a serverom cez sprostredkovateľa pôsobí sprostredkovateľ **vo vzťahu ku klientovi ako server a vo vzťahu k serveru ako klient**. Mnohí sprostredkovatelia sú navrhnutí tak, aby boli schopní odpočúvať rôzne protokoly TCP/IP. Väčšina protokolov nevie o existencii vloženého sprostredkovateľa. Protokol HTTP však obsahuje špeciálnu podporu pre určitých sprostredkovateľov, ako sú **proxy servery**, ktorým poskytuje nástroje (hlavičky) na narábanie so žiadosťami a odpoveďami HTTP.

Protokol HTTP - Proxy

- Protokol HTTP zavádza **proxy, bránu a tunel**. Jedná sa o medziľahlé systémy, ktoré môžu ležať na ceste medzi klientom a serverom. Na ceste od klienta k serveru môže ležať ľubovoľné množstvo týchto medziľahlých systémov. Z hľadiska protokolu TCP sa nadväzuje TCP spojenie vždy mezi dvomi uzlami. Tj. TCP spojenie medzi klientom a prvým proxy, medzi prvým proxy a druhým proxy atď. Pri opise proxy, brány a tunelu sa obmedzíme najprv na situáciu, že medzi klientom a serverom je iba jeden medziľahlý systém. Následne si potom povieme, že umiestením viacero medziľahlých systémov sa vôbec nič nezmení.
- Najčastejšie sa preto tieto systémy používajú tam, kde nie je možné priamo nadviazať TCP spojenie mezi klientom a serverom. Tj. napr. na bezpečnostnej bráne oddeľujúcej intranet od Internetu.
- Proxy je systém skladajúci sa z dvoch častí:
  - Zo serverovej časti, ktorá prijíma požiadavky klienta akoby ich prijímal cieľový server. Požiadavky však v zápätí odovzdá klientskej časti.
  - Z klientskej časti, ktorá prevezme požiadavky od serverovej časti, nadviaže TCP spojenie s cieľovým serverom a odovzdá menom klienta požiadavky cieľovému serveru na vybavanie.

Protokol HTTP - Proxy

- Takto sa Proxy javí používateľovi. Avšak uprostred Proxy medzi serverovou a klientskou časťou je ešte skrytá **vlastná logika Proxy**. Proxy totiž rozumie aplikačnému protokolu (v našom prípade protokolu HTTP) a s prijatou požiadavkou od klienta môže vykonať niekoľko operácií:
  - Môže prepísať požiadavku (resp. odpoveď), tj. zmeniť údaje aplikačného protokolu.
  - Odpovedi môže ukladať do pamäti cache (napr. na disk). Pokiaľ proxy obdrží v budúcnosti rovnakú požiadavku (napr. i od iného klienta), potom môže vrátiť túto požiadavku rýchlejšie priamo z pamäti bez toho, aby nadväzoval spojenie s cieľovým serverom. Vypadá to síce efektívne, ale zásadnou otázkou takto uchovávaných odpovedí je ich aktuálnosť. Ukladanie dat do pamäti sa tak stáva komplikovanou záležitosťou, ktorej sa venuje samostatný odstavec. V dnešnej dobe už máme k dispozícii veľmi sofistikované algoritmy na prácu s pamäťou cache, ale i vďaka používaniu dynamických stránok, stavových transakcií a zabezpečovaných spojení a v neposlednom rade zvýšeniu priepustnosti komunikačních liniek začína význam pamäti cache na Proxy ustupovať.
  - Môže zisťovať či klient je oprávnený takú požiadavku vykonať.
- Proxy **môže preverovať** oprávnenosť klienta vykonať nejakú požiadavku z niekoľkých hľadísk:
  - Môže zisťovať, **či klient nepristupuje na nejaký nežiadúcí server**. Napr. zamestnávateľ môže nechať na proxy nastaviť zoznam serverov, na ktoré si nepraje, aby jeho zamestnanci pristupovali. V praxi je to bežné, že zamestnávateľ zakáže prístup napr. na www.playboy.com (zamestnanci si ale potom na Internete nájdu 10 iných serverov s pre nich ešte zaujímavejšou tématikou, o ktorých zamestnávateľ nevie).

Protokol HTTP - Proxy

- Proxy **môže preverovať** oprávnenosť klienta vykonať nejakú požiadavku z niekoľkých hľadísk:
  - Môže zisťovať, **či používateľ je oprávnený proxy používať**. V takomto prípade vyžaduje autentizáciu používateľa. Nejčastejší typy autentizácie používateľa sú:
    - Pomocou IP-adresy stanice, na ktorej používateľ pracuje. Táto autentizácia nie je príliš bezpečná, preto slúži skorej k administratívnemu obmedzeniu niektorých klientov nepoužívať proxy (napr. nepristupovať cez proxy do Internetu).
    - Pomocou mena používateľa a hesla.
    - Pomocou mena používateľa a jednorázového hesla.
  - Proxy bežiaci na bezpečnostnej bráne môže od operačného systému požadovať, aby vykonával kontrolu **z akého sieťového rozhrania prichádza používateľova požiadavka na proxy**. Tj. či používateľ pristupuje zo sieťového rozhrania vnútornej siete, či zo sieťového rozhrania do Internetu. To pochopitelne štandardná implementácia TCP/IP v operačnom systéme nevie. Čo býva práve jedným z dôvodov prečo bezpečnostné brány pri svojej inštalácii zásadne zasahujú do operačného systému. Podľa týchto informácií potom proxy zisťuje, či je to požiadavka z intranetu, či je to napr. útok z Internetu. Útokom z Internetu môže byť tiež bránené tým, že serverová časť proxy počúva iba IP-adresy vnútornej sieti proxy.
  - V prípade, že proxy vie odkiaľ požiadavka prišla (či z vnútornej siete alebo z Internetu), potom **môže použiť iný autentizačný mechanizmus** na požiadavky z vnútornej siete a na požiadavky z Internetu. Napr. z vnútornej siete vybavuje všetky požiadavky, kdežto z Internetu vyžaduje autentizáciu jednorázovým heslom.
  - I proxy môže data predtým než ich odovzdá (i uloží do cache) **skontrolovať, či neobsahujú vírusy**. Väčšinou to nerobí samotné proxy, ale proxy volá iný proces, ktorý vykoná túto špeciálnu kontrolu (zvyčajne na inom špecializovanom počítači).

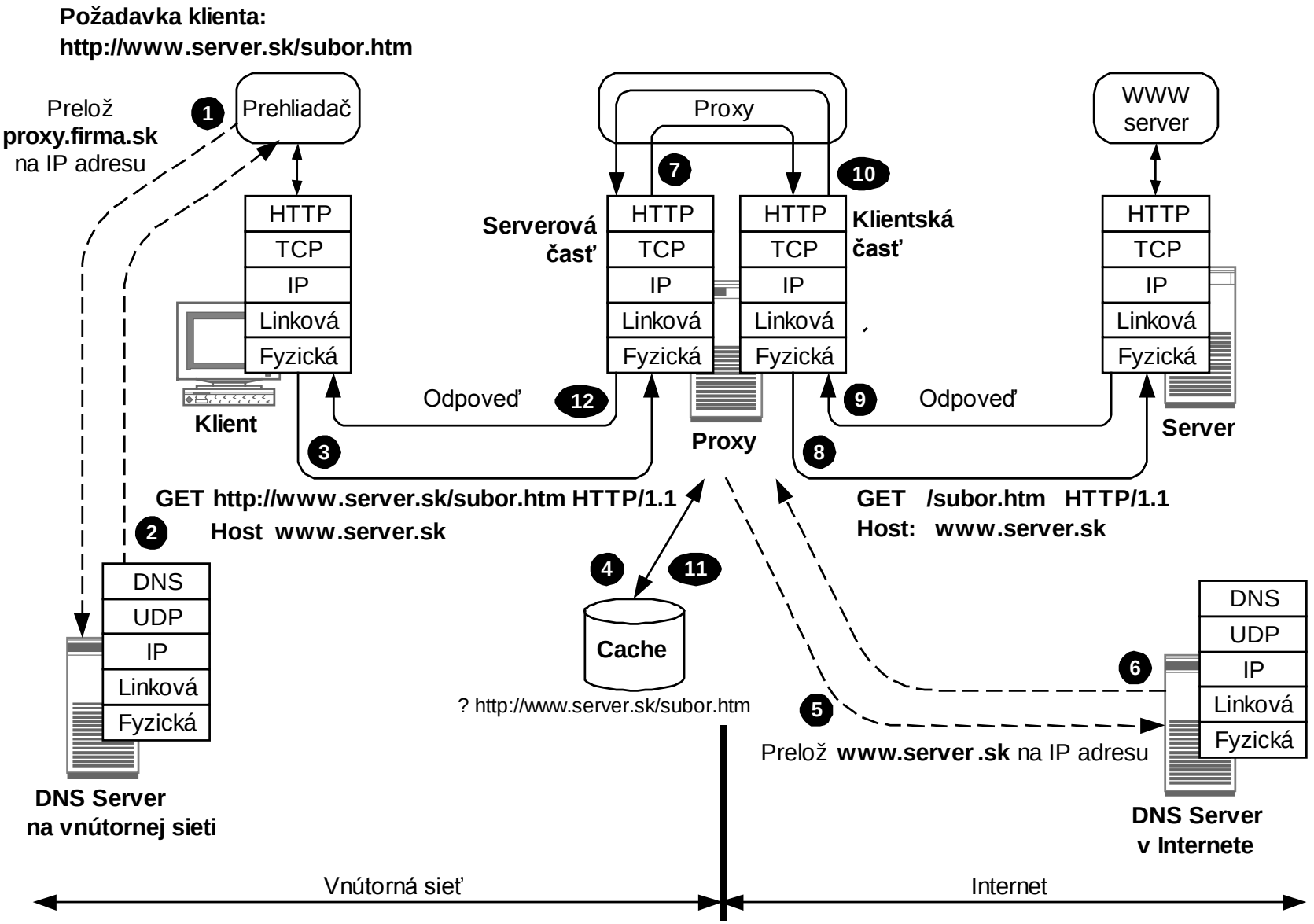
Protokol HTTP - Proxy

- Na nasledujúcom obrázku je schématicky znázornená činnosť proxy. Na začiatku používateľ zapíše do okna svojho prehliadača identifikátor objektu (URI), ktorý chce prehliadať. Napr. klient do okna prehliadača vloží požiadavku: **http://www.server.sk/subor.htm**
- Klient bude však vybavovať túto požiadavku cez proxy. Tj. v konfigurácii svojho prehliadača zapísal meno proxy, prostredníctvom ktorého sa bude požiadavka vybavovať. Prehliadač z identifikácie objektu zistí iba aplikačný protokol. Ako sa je možné presvedčiť pri nastavení prehliadača, tak pre každý protokol môže klient použiť iné proxy.
  - V prvom kroku klient preloží meno proxy na IP-adresu (1 a 2). Meno cieľového servera totiž nemusí byť v intranete ani preložiteľné. Teraz klient nadviaže TCP komunikáciu so serverovou časťou proxy na porte uvedenom v konfigurácii klienta. Do takto vytvoreného TCP spojenia vloží klient svoju HTTP požiadavku (3):
    GET http://www.server.sk/subor.htm HTTP/1.1
    Host:www.server.sk
  - Proxy vo svojej pamäti cache preverí, či odpoveď na túto požiadavku náhodou nemá k dispozícii (4).
  - V prípade, že požiadavka v pamäti cache nebola nájdená, tak odovzdá požiadavku klientskej časti na vybavenie. Klientská časť musí z URI požiadavky vybrať meno servera (www.server.sk) a preložiť ho v DNS (5 a 6). Pretože proxy má už prístup do Internetu, tak je už schopný túto požiadavku nechať preložiť v Internete.
  - Klientská časť proxy najprv prepíše požiadavku na
    GET /soubor HTTP/1.1
    Host: www.server.sk

Protokol HTTP - Proxy

- Následne klientská časť nadviaže spojenie s cieľovým serverom protokolom TCP a odovzdá mu požiadavku (8) menom klienta. Server vráti odpoveď (9), ktorú obdrží proxy (10). Pokiaľ je odpoveď prípustná uložiť do pamäti cache, potom ju tam uloží (11). Proxy odovzdá odpoveď klientovi (12), ktorý ju zobrazí používateľovi a prípadne si ju tiež uloží do svojej pamäti cache.

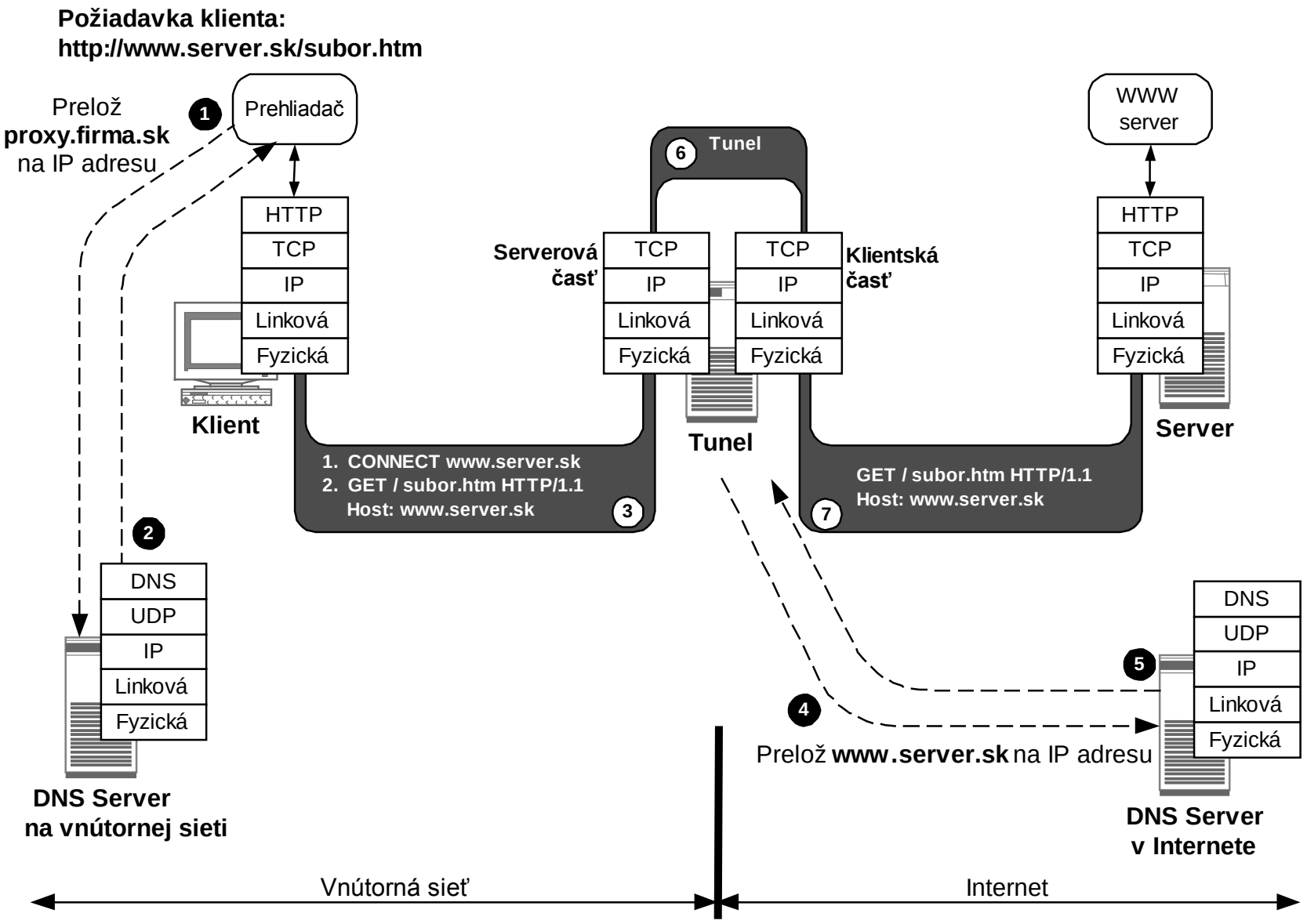
Protokol HTTP - Proxy



Protokol HTTP – Brána a Tunel

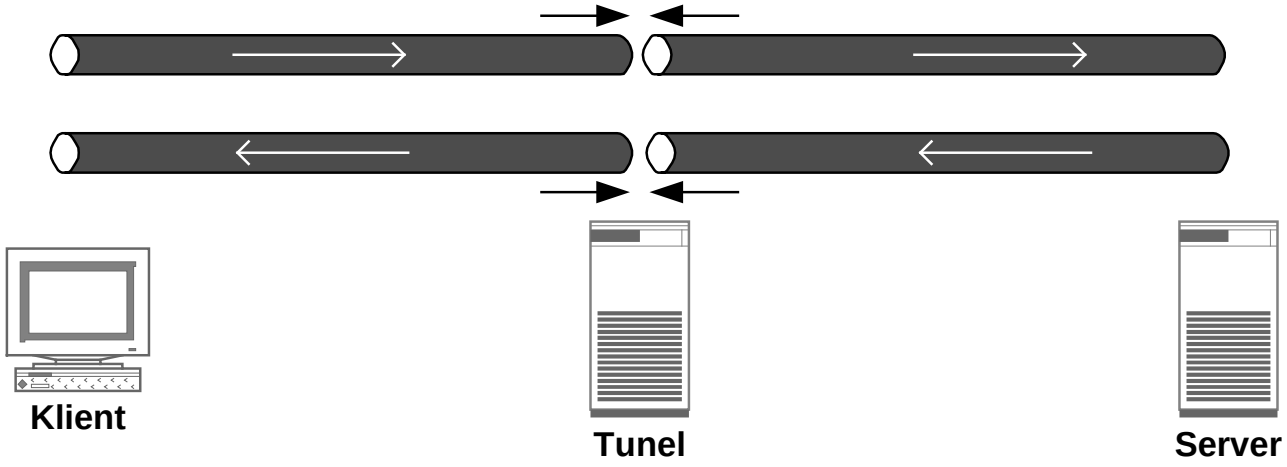
- **Brána (gateway)** – je medziľahlý uzol, ktorý pracuje ako proxy s tým rozdielom, že mení aplikačný protokol. Najbežnejším prípadom je keď serverová časť klienta prijíma požiadavku protokolu HTTP a mení ju na komunikáciu v protokole FTP.
- **Tunel** je trochu iná koncepcia. Tunel totiž nemusí „rozumieť“ prenášaným datam. Tunelem je možné dokonca prenášať aplikačné data zašifrované. Toho využíva protokol SSL.
- Činnosť tunelu si objasníme na nasledujúcom obrázku. Klient preloží meno tunelu na IP adresu (1 a 2). Klient nadviaže s tunelom TCP spojenie. Do takto vytvoreného kanálu klient spravidla vloží iba príkaz (tj. metódu) CONNECT s menom a portom cieľového serveru (3). Tunel preloží meno cieľového servera na IP adresu (5 a 6) a nadviaže s cieľovým serverom TCP spojenie na porte uvedenom v príkaze CONNECT.

Protokol HTTP - Tunel



Protokol HTTP - Tunel

- Teraz má tunel nadviazané dve obojsmerné spojenia. Každé spojenie si predstavíme ako dve rúry (pozri na obrázok nižšie): jedna rúra je na spojenie tam a druhá na spojenie späť (duplexný spoj).
- Tunel neurobí nič iného ako rúry „navarí na seba“, tj. tunel bez toho aby vedel čo prenáša, tak mechanicky čo príde od klienta odovzdá do rúry na cieľový server. Podobne všetko, čo príde zo servera odovzdá klientovi.
- V takomto spojení potom klient môže začať napr. protokolom SSL nadväzovať šifrované spojenie.
- Je celkom pochopiteľné, že tunel nevidí do prenášaných dat, takže nemôže kontrolovať čo klient za data zo servera sťahuje. Či sa napr. nejedná o Java applety alebo ActiveX komponenty.
- Po ukončení spojenia sa celý tunel rozpadá.

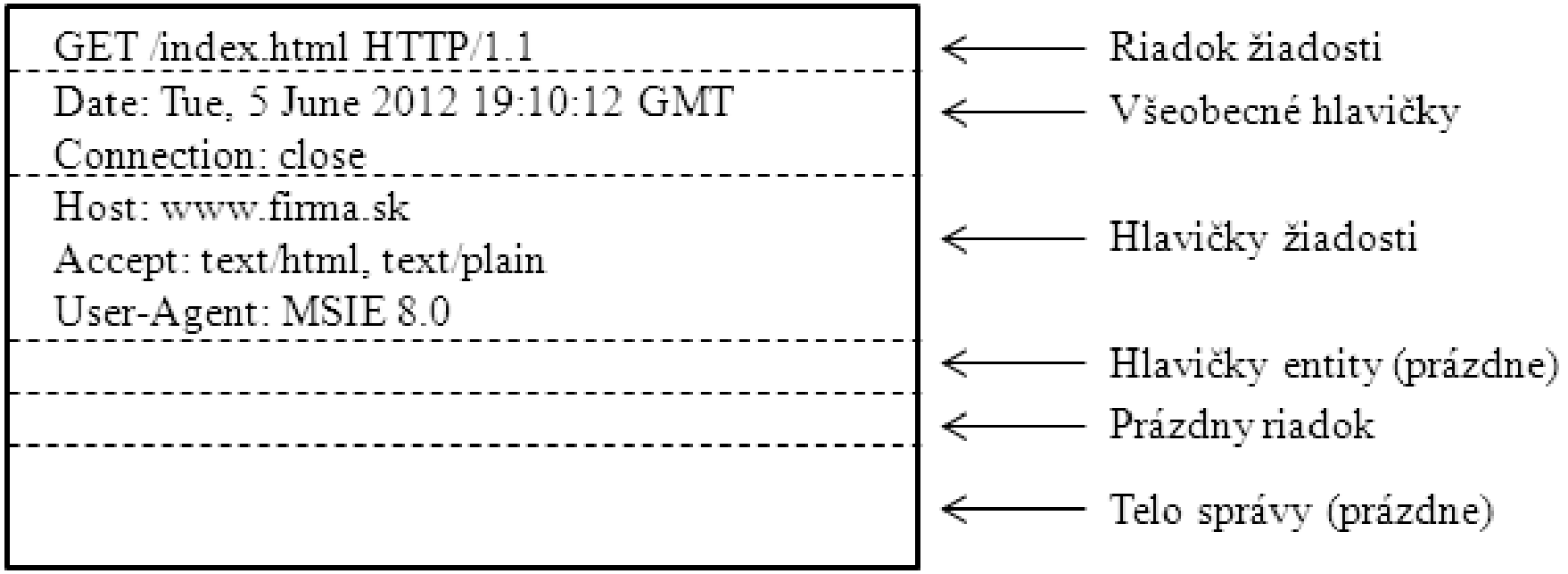


Protokol HTTP - Dopyt

- **HTTP dopyt**
  - Štruktúra HTTP dopytu (i odpovedi) pripomína štruktúru e-mailu. Na prvý pohľad vidíme rozdiel iba v prvom riadku. Prvý riadok dopytu obsahuje tzv. metódu a prvý riadok odpovedi obsahuje tzv. stavový riadok.
  - HTTP dopyt sa skladá (viď príklad na nasledujúcom obrázku) z:
    - **Metódy.** Protokol HTTP verzia 1.1 podporuje tieto metódy: GET, POST, HEAD, OPTIONS, TRACE, CONNECT, PUT a DELETE. Metódy PUT a DELETE nebývajú implementované.
    - **Záhlavie**, ktoré je tvorené jednotlivými hlavičkami. **Každá hlavička začíná kľúčovým slovom** (napr. Host:). Kľúčové slovo je ukončené dvojbodkou nasledovanou medzerou. Za medzerou potom môžu nasledovať parametre hlavičky. Celá hlavička je vždy ukončená koncom riadku CRLF. Iba jediná hlavička je povinná, a to hlavička **Host**.
    - **Prázdneho riadku** (CRLF CRLF, kde prvé CRLF ukončuje posledný riadok hlavičky)
    - **Prenášaných údajov** (voliteľne).
  - Metóda má v protokole HTTP verzia 1.1 vždy tvar:
    - `<Názov metódy> <URI> HTTP/1.1`
    - kde je verzia protokolu HTTP/1.1, čo je t.č. aktuálna verzia. Uvedenie verzie je povinné. Pokiaľ by sme uviedli iba názov metódy a URI (Uniform Resource Identifier), potom by sa jednalo o dopyt v protokole HTTP verzie 0.9. Nesmieme sa potom diviť, že v odpovedi nedostaneme ani stavový riadok, ten bol totiž zavedený až vo verzii 1.0 (verzia 1.0 nemala zase okrem iného hlavičku Host:).

Protokol HTTP - Dopyt

- Na obrázku sú ukázané elementy štruktúry formátu správy žiadosti HTTP a príklad typov hlavičiek, ktoré by mohla obsahovať. Podobne ako väčšina žiadostí HTTP ani táto nenesie žiadnu entitu, takže neexistujú žiadne hlavičky entity a telo správy je prázdne.



Protokol HTTP – Dopyt, metóda GET

- **Metóda GET**
  - Metóda GET slúži na dopytovanie klienta na konkrétne informácie uložené na serveri. Dopyt je formulovaný ako súčasť URI v reťazci <dopyt> (tj. v reťazci za otáznikom). Metóda GET môže teoreticky obsahovať i data, avšak táto možnosť býva zriedka využívaná (data sa prenesú ako súčasť dopytu).
  - Príklady budeme demonštrovať pomocou programu telnet vo Windows 2000. Ako server použijeme info.firma.sk bežiaci na porte 80:
    C:\> telnet info.firma.sk 80
  - riadky, ktoré sú napísané z klávesnice sa uvádzajú reťazcom „C: “ a riadky, ktoré odpovedal server sú uvádzané „S: “. Prvým príkladom je požiadavka na výpis obsahu koreňového adresáre servera (adresára /). Webové servery bývajú nakonfigurované tak, že väčšinou pri požiadavke na obsah adresára nevracajú výpis súborov, ktoré adresár obsahuje, ale súbor menujúci sa spravidla index.html a obsahujúci nejakú správu. Takže v koreňovom adresári bude tento súbor obsahovať titulnú stránku webového servera.

Protokol HTTP – Dopyt, metóda GET

- ***Príklad:***
  - C: GET / HTTP/1.1
  - C: Host: info.firma.sk
  - C: Prázdny riadok ukončujúci záhlavie
  - S: HTTP/1.0 200 OK
  - S: Date: Wed, 20 Dec 2000 17:20:18 GMT
  - S: Last-Modified: Mon, 10 Apr 2000 07:14:43 GMT
  - S: Content-Length: 2855
  - S: Server: Apache/1.2b10
  - S: ETag: "e9ed-b27-38f17f63"
  - S: Accept-Ranges: bytes
  - S: Content-Type: text/html
  - S: Prázdny riadok ukončujúci záhlavie odpovedi
  - S: <HTML>
  - S: <HEAD>
  - S: <TITLE>info.firma.sk </TITLE>
  - S: </HEAD>
  - S: <TABLE>
  - S: <TR><TD> Vítame Vás na servere ….

- V protokole HTTP verzia 1.1 je povinná hlavička Host, preto i náš dopyt musel túto hlavičku obsahovať. Hlavička Host: obsahuje meno servera.

Protokol HTTP – Dopyt, metóda GET

- Odpoveď obsahuje niekoľko zaujímavých hlavičiek:
  - **Date:** Táto hlavička obsahuje dátum a čas, kedy bola odpoveď vytvorená.
  - **Last-Modified:** Táto hlavička obsahuje čas poslednej modifikácie zdroja (webovej stránky).
  - **Content-Lenght:** Táto hlavička obsahuje dĺžku údajov odpovedi v bajtoch.
  - **Accept-Ranges:** Táto hlavička obsahuje informáciu o tom, že server akceptuje dopyty, ktoré sú zaslané v niekoľkých častiach. Pričom veľkosť časti sa udáva v bajtoch.
  - **ETag:** Jednoznačne identifikuje verziu odpovedi. Pokiaľ je v pamäti cache odpoveď rovnakej verzii ako na servere, potom sú obidve odpovedi zhodné.
- Metóda GET môže dávať tiež podmienené dopyty využitím hlavičiek **If-Modified-Since**, **If-Unmodified-Since**, **If-Match**, **If-None-Match** a **If-Range**. Tieto dopyty umožňujú prenášať data odpovedi iba v prípade, že podmienka v dopyte je pravdivá. Hlavičky If-Match a If-None-Match vyhodnocuje identifikácia verzie odpovedi (ETag), kdežto hlavičky If-Modified-Since a If-Unmodified-Since vyhodnocuje dátum poslednej modifikácie. Hlavička If-Range vyhodnocuje buď identifikácia alebo verzia odpovedi, požaduje však od servera len tú časť odpovedi, ktorá bola zmenená.

Protokol HTTP – Dopyt, ďalšie metódy

- **Metóda POST**
  - Metódou POST odosielame na server data (napr. pole HTML-formulára).
- **Metóda HEAD**
  - Metóda HEAD požaduje iba hlavičku odpovedi bez dat.
  - ***Príklad:***
    - C: HEAD / HTTP/1.1
    - C: Host: info.firma.sk
    - C:
    - S: HTTP/1.0 200 OK
    - S: Date: Wed, 20 Dec 2000 17:20:18 GMT
    - S: Last-Modified: Mon, 10 Apr 2000 07:14:43 GMT
    - S: Content-Length: 2855
    - S: Server: Apache/1.2b10
    - S: ETag: "e9ed-b27-38f17f63„
    - S: Accept-Ranges: bytes
    - S: Content-Type: text/html
- **Metóda TRACE**
  - Je akousi obdobou príkazu **tracert**. Tentoraz však nezistíme, koľko je medzi naším a cieľovým počítačom smerovačov, ale koľko je tam proxy alebo brán.
  - V prípade, že komunikujeme s proxy alebo bránou, tak nesmieme zabudnúť, že do metódy (TRACE) musíme zapísať celé absolútne URI (http://info.firma.sk).

Protokol HTTP – Dopyt, ďalšie metódy

- ***Príklad:***
  - C: TRACE http://info.firma.sk HTTP/1.1
  - C: Host: info.firma.sk
  - C:
  - S: HTTP/1.0 200 OK
  - S: Date: Wed, 20 Dec 2000 17:24:04 GMT
  - S: Server: Apache/1.2b10
  - S: Content-Type: message/http
  - S:
  - S: TRACE / HTTP/1.0
  - S: Host: info.firma.sk
  - S: Cache-Control: Max-age=259200
  - S: Via: 1.1 proxy.firma.sk:8080 (Squid/1.1.22)
- V údajovej časti odpovedi je potom štatistika, ktorá nás zaujíma, uvedená v hlavičke Via. V tejto hlavičke sú uvedené jednotlivé proxy alebo brány oddelené čiarkou. V našom prípade sme komunikovali iba cez jeden proxy.
- O každom proxy či bráne môžu byť uvedené štyri údaje:
  - **Protokol**, ktorý v prípade protokolu HTTP môže chýbať.
  - **Verzia** protokolu (1.1).
  - **Medziľahlý systém** (proxy alebo brána – v našom príklade proxy.firma.sk:8080).
  - **Komentár** v guľatých zátvorkách obsahujúci spravidla informácie o softvére medziľahlého systéme (Squid/1.1.22).
- Pokiaľ by na ceste bolo viacej medziľahlých systémov, potom by za uvedeným systémom nasledovala čiarka a ďalší systém atď.

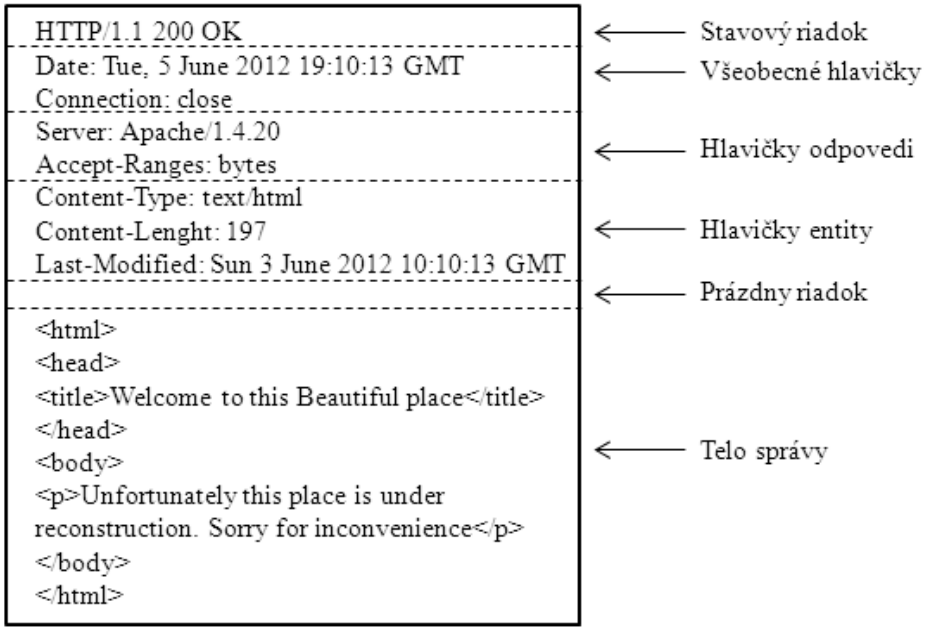
Protokol HTTP – Dopyt, ďalšie metódy

- Zaujímavá hlavička je **Cache-Control**, ktorá v našom prípade udáva, že informácie majú byť udržované v pamäti cache maximálne 259200 sukúnd.

- **Metóda OPTIONS**
  - Metódu OPTIONS používa na zistenie komunikačných vlastností servera či požadovaného URI. V prípade, že sa dopytujeme na vlastnosti celého servera, potom namiesto URI použijeme hviezdičku:
  - ***Príklad:***
    - C: OPTIONS * HTTP/1.1
    - C: Host: info.firma.sk
    - C:
    - S: HTTP/1.1 200 OK
    - S: Date: Sat, 23 Dec 2000 19:11:22 GMT
    - S: Server: Apache/1.2b10
    - S: Content-Length: 0
    - S: Allow: GET, HEAD, OPTIONS, TRACE
  - Server nám oznamuje, že podporuje metódy GET, HEAD, OPTIONS a TRACE.

Protokol HTTP – Odpoveď

- **HTTP odpoveď**
  - HTTP-odpoveď začína stavovým riadkom, ktorý má tvar: <Verzia> <Výsledkový kód> <Poznámka>, Kde verzia je verziou protokolu HTTP, v ktorej je odpoveď formulovaná, Výsledkový kód špecifikuje úspešnosť/neúspešnosť operácie a poznámka textovo objasňuje výsledok operácie. Za stavovým riadkom nasleduje opäť záhlavie tvorené hlavičkami. Záhlavie je ukončené prázdnym riadkom, ktorý oddeľuje záhlavie od prenášaných dat.
  - Príklad stavového riadku (kladná odpoveď): HTTP/1.1 200 OK
  - Za stavovým riadkom, podobne ako v prípade HTTP-dopytu, začína záhlavie tvorené hlavičkami. Za záhlavím nasleduje prázdny riadok. Za prázdnym riadkom môžu nasledovať data odpovedi.



Protokol HTTP – Odpoveď

- Výsledkové kódy sú trojciferné. Prvá cifra charakterizuje odpoveď:
  - **1xx je informačná správa**, ktorá poskytuje všeobecnú informáciu, neindikuje úspešné vykonanie žiadosti HTTP alebo chybu
  - **2xx je správa o úspešnom vykonaní žiadosti HTTP**, ktorá indikuje, že matóda uvedená v žiadosti bola serverom prijatá, pochopená a akceptovaná.
  - **3xx je správa o presmerovaní**, ktorá indikuje, že žiadosť priamo nezlyhala, ale je potrebná dodatočná akcia predtým než bude žiadosť úspešne vykonaná.
  - **4xx je správa o chybe klienta**, ktorá indikuje, že žiadosť HTTP je neplatná, obsahuje zlú syntax alebo nemôže byť vykonaná z nejakých iných dôvodov pre chybu klienta.
  - **5xx je správa o chybe servera**, ktorá indikuje, že žiadosť HTTP je platná, ale server nebol schopný ju vykonať z dôvodu jeho vlastného problému. Na Obrázku č. 10.21 sú uvedené stavové kódy aj s frázou dôvodu podľa [RFC 2616].
- Pokiaľ sa informácia nevmestí do stavového riadku, potom je do odpovedi pridaná ďalšia informácia v hlavičke **Warning**. Ide v podstate o doplňujúcí stavový riadok. Hlavička Warning má dva parametre oddelené medzerou: výsledkový kód a poznámku.
  - Nejčastejšie sa hlavička Warning používa k doplneniu informácií podávaných z pamäti cache, tj. nie z prvej ruky (zo serveru). Môže sa totiž stať, že cache vracia prešlé informácie, pretože napr. proxy nie je schopný nadviazať spojenie ďalej smerom k serveru (to sú výsledkové kódy 110 až 112).
  - Prehľad výsledkových kódov používaných v hlavičke Warning: 110 Response is stale, 111 Revalidation failed, 112 Disconnected operation, 113 Heuristic expiration,199 Miscellaneous warning

Protokol HTTP – Odpoveď, stavové kódy

Stavový kód	Význam (Fráza dôvodu)	Stavový kód	Význam (Fráza dôvodu)
100	Continue	404	Not Found
101	Switching Protocols	405	Method not Allowed
200	OK	406	Not Acceptable
201	Created	407	Proxy Authentication Required
202	Accepted	408	Request Timeout
203	Non-Authoritative Information	409	Conflict
204	No Content	410	Gone
205	Reset Content	411	Length Required
206	Partial Content	412	Precondition Failed
300	Multiple choices	413	Request Entity Too Large
301	Moved Permanently	414	Request-URI Too Long
302	Found	415	Unsupported Media Type
303	See Other	416	Requested Range Not Satisfiable
304	Not Modified	417	Expectation Failed
305	Use Proxy	500	Internal Server Error
306	(Unused)	501	Not Implemented
307	Temporary Redirect	502	Bad Gateway
400	Bad Request	503	Service Unavailable
401	Unauthorised	504	Gateway Timeout
402	Payment Required	505	HTTP Version Not Supported
403	Forbidden		

Protokol HTTP – ostatné hlavičky

- **Hlavičky Accept**
  - Hlavičky **Accept, Acept-Charset, Accept-Encoding a Accept-Language** sú hlavičky ktorými klient vo svojom dopyte oznamuje svoje možnosti. Každá z týchto hlavičiek môže obsahovať niekoľko eventualít oddelených čiarkou. Pri každej eventualite môže byť za stredníkom uvedená kvalita (q). Kvalita je číslo mezi 0 a 1. Čím vyššie má eventualita kvalitu, tým viacej ju klient preferuje (implicitne sa predpokladá q=1). Na špecifikáciu eventuality je možné uviesť hviezdičku označujúcu všetky možnosti.
  - Hlavičkou **Accept** klient špecifikuje podporované typy médií (viď hlavička Content-Type):
    - Accept: text/*;q=0.3, text/html, image/jpeg;q=0.7, model/vrml, */*;q=0.1
    - Hovorí, že klient uprednostňuje: ľubovolný text s kvalitou 0.3, text/html s kvalitou 1, image/jpeg s kvalitou 0.7, model/vrml s kvalitou 1, ľubovolné médium s kvalitou 0.1.
  - Hlavičkou **Accept-Charset** klient špecifikuje podporované znakové sady:
    - Accept-Charset: iso-8859-5, unicode-1-1;q=0.8, *;q=0.1
    - Hovorí, že klient uprednostňuje znakovú sadu iso-8859-5 (s kvalitou 1), ďalej s kvalitou 0.8 podporuje znakovú sadu unicode-1-1. Inak podporuje ľubovoľnú znakovú sadu s kvalitou 0.1.
  - Hlavičkou **Acceept-Encoding** klient oznamuje podporované typy kódovania dat:
    - Accept-Encoding: compress;q=0.5, gzip
    - Klient preferuje metódu gzip, avšak s kvalitou 0.5 podporuje i metódu compress.
  - Hlavičkou **Accept-Language** klient oznamuje podporované jazyky:
    - Accept-Language: sk, en;q=0.5
    - Klient preferuje slovenčinu s kvalitou 1, ale podporuje i angličtinu s kvalitou 0.5.
  - Hlavička **Accept-Ranges** je používaná v odpovedi servera klientovi. Viď prvý príklad.

Protokol HTTP – Autorizácia klienta

- **Autorizácia klienta**
  - Klient síce môže priamo do URI zapísať meno používateľa a heslo, to je však málo bežné. Bežnejší je dialóg, kedy klient nezadá svoje autentizačné informácie a server vráti:
    HTTP/1.1 401 Unauthorized
    WWW-authenticate: autent_metóda realm="reťazec", prípadné_ďalšie_parametre
    - Kde prvý parametr je typ autentizačnej metódy, ktorú server vyžaduje.
    - Reťazec „realm" bude zobrazený klientovi, aby vedel k akému objektu sá má autentizovať.
    - Konečne niektoré autentizačné metódy môžu používať ďalšie parametre. Autentizačná metóda Basic napr. ďalšie parametre nepoužíva.
  - RFC-2617 rozoznáva dva typy autentizácie:
    - Basic, kde autentizačná metóda je „Basic".
    - Digest, kde autentizačná metóda je „Digest".
  - Obidve metódy používajú autentizáciu menom používateľa a heslom.
  - Autentizácia Basic prenáša sieťou meno a heslo v textovom (nezabezpečenom tvare). Autentizačný dialog potom prebieha napr.:
    - C: GET /subor HTTP/1.1
    - C: Host: info.firma.sk
    - C:
    - S: HTTP/1.1 401 Unauthorized
    - S: WWW-authenticate: Basic realm="info.firma.sk"
    - S: …ďalšie hlavičky

Protokol HTTP – Autorizácia klienta

- C: GET /soubor HTTP/1.1
- C: Host: info.firma.sk
- C: Authorization: Basic RG9zdGFsZWs6aGVzbG8=
- C:
- S: HTTP/1.1 200 OK
- S: …

- Kde klient po obdržaní „HTTP/1.1 401 Unauthorized“ vykoná autentizáciu menom a heslom. Lenže server môže ponúkať väčšie množstvo objektov a ku každému z nich môžeme použiť inú autentizáciu. Preto server vracia reťazec „realm“, aby prehliadač mohol používateľovi do dialógového okna zobraziť k akému objektu má zadať meno používateľa a heslo. Z mena a hesla je vytvorený reťazec. Meno a heslo sú oddelené dvojbodkou (napr. JanBiely:heslo). V hlavičke Authorization sa neprenáša reťazec priamo, ale kódovaný Base64, tj. RG9zdGFsZWs6aGVzbG8=.
- Komukoľvek, kto na ceste od klienta k serveru odchytí hlavičku Authorization, tak na reťazec RG9zdGFsZWs6aGVzbG8= stačí pustiť dekódovanie Base64 (napr. programom ssleay) a získa heslo.
- Tomu se snaží zabrániť autentizácia typu Digest. Tento typ autentizácie rovnako používa heslo používateľa, avšak neprenáša heslo samotné, ale kontrolný súčet (počítaný napr. algoritmom MD5) z:
  - **Čísla „nonce“** generovaného serverom ako kontrolný súčet z: časovej pečiatky, identifikácie odpovedi (ETag) a súkromného kľúč servera.
  - **Čísla „opaque“** generovaného rovnako serverom. Čísla „nonce“ a „opaque“ sú odovzdané klientovi v hlavičke WWW-authenticate ako ďalšie parametre.
  - **Mena a hesla používateľa,** Reťazca z parametra „realm“, Požadovaného URI.
- Ďalšie možnosti autentizácie prináša RFC-2831. (Autentizácia Basic, avšak umožňuje používať jednorázové heslo generované autentizačným kalkulátorom.)

Protokol HTTP – Autentizácia proxy

- **Autentizácia proxy**
  - Autentizácia klienta voči proxy je celkom podobná autentizácii klienta voči serveru. V prípade, že proxy vyžaduje autentizáciu, potom v odpovedi:
    407 Proxy Authentication Required
    Proxy-authenticate: autent_metóda realm=“reťazec“,
    prípadné_ďalšie_parametre
  - Požaduje od klienta autentizáciu. Klient sa autentizuje použitím hlavičky Proxy-Authorization. Hlavičky Proxy-Authenticate a Proxy-Authorization majú rovnakú syntax ako hlavičky WWW-Authorization a Authenticate.

Protokol HTTP – Hlavičky Content

- **Hlavičky Content**
  - Hlavičky Content vychádzajú z filozofie MIME, avšak so samotnými MIME nie sú celkom kompatibilné. Protokol HTTP nepodporuje napr. hlavičku Content-Transfer-Encoding a pochopiteľne ani hlavičku Mime-Version.
  - Hlavička **Content-Type** je obdobou tejto hlavičky v MIME. Opisuje typ média prenášaných dat. Napr.: Content-Type: text/html; charset=ISO-8859-4. Špecifikuje, že prenášané data sú text formátovaný v jazyku HTML. Použitá znaková sada je ISO-8859-4.
  - Hlavička **Content-Length** obsahuje dĺžku prenášaných dat.
  - Hlavička **Content-Encoding** špecifikuje kódovací algoritmus použitý pred odoslaním dat. Pretože protokol HTTP je osembitový, tak sa kódovaním nerozumie napr. base64, ale napr. kompresia dat. Príklad: Content-Encoding: gzip
  - Hlavička **Content-Language** špecifikuje jazyk. Napr. Content-Language: sk
  - Hlavička **Content-MD5** obsahuje kontrolný súčet algoritmom MD-5 z prenášaných dat.
  - Hlavička **Content-Range** sa použije v prípade, že správa obsahuje iba časť prenášaných dat. Napr. odpoveď servera je príliš veľká, tak server svoju odpoveď rozkúskuje na niekoľko odpovedí:
    - S: HTTP/1.1 206 Partial content
    - S: Content-Range: bytes 21010-47021/47022
    - S: Content-Length: 26012
    - S: …
    - (za lomítkom je celková dĺžka správy)

Protokol HTTP – Hlavičky Content

- **Hlavičky Content**
  - Hlavička **Content-Location** obsahuje URI s prenášanými datami. Táto hlavička má význam najmä v prípade, kedy požadované data sú na niekoľkých lokalitách. Táto hlavička nesúvisí s presmerovaním! Podobný význam má hlavička **Referer**, ktorou môže klient serveru oznámiť odkiaľ klient získal informácie o požadovanom URI.

Protokol HTTP – Presmerovanie a dočasná nedostupnosť

- **Presmerovanie a dočasná nedostupnosť**
  - Môže sa stať, že požadovaný objekt bol premiestnený na iné URI (napr. na iný server či do iného adresára). V takomto prípade server vráti stavový riadok s výsledkovým kódom 3xx a hlavičkou: Location: nové-URI. Napr.
    HTTP/1.1 301 Moved Permanently
    Location: http://www.firma.sk/subor
    …
  - Avšak môže sa i stať, že objekt nebol presmerovaný, ale je dočasne nedostupný, potom môže server klientovi oznámiť nielen zlú správu, že je objekt nedostupný, ale pomocou hlavičky **Retry-After** môže klientovi poradiť za ako dlho sa má pokúsiť znovu na objekt pýtať. Napr. mu poradí, aby sa pýtal za 1 minutu:
    HTTP/1.1 503 Service Unavailable
    Retry-After: 60
    …
  - Hlavička Retry-After môže mať i význam v prípade presmerovania:
    HTTP/1.1 301 Moved Permanently
    Location: http://www.firma.sk/subor
    Retry-After: 60
    …
    V tomto prípade, server oznamuje klientovi, aby presmerovanie vykonal až za 60 sekúnd.

Protokol HTTP – Hlavička Upgrade

- **Hlavička Upgrade**
  - Hlavičkou Upgrade klient oznamuje serveru, že by chcel v existujúcom TCP spojení prejsť na „lepší“ protokol. Tj. napr. protokol novšej verzie či protokol bezpečnejší. Napr.: Upgrade: HTTP/2.0
  - Server potvrdzuje prechod do uvedeného protokolu stavovým kódom „101 Switching Protocols“

Protokol HTTP – Bezpečnosť a privátnosť

- Protokol HTTP priamo neobsahuje žiadny mechanizmus **na ochranu privátnosti** prenášaných dokumentov alebo správ. Existujú však dva rôzne prístupy, ktorými sa to zvyčajne zabezpečuje.
  - Najjednoduchší spôsob je **šifrovanie zdroja na serveri** a dodanie platného dešifrovacieho kľúča iba oprávneným používateľom. Aj keď útočník zachytí celú správu, entita sama bude stále zabezpečená.
  - Ďalšou bežnejšou metódou je **použitie dodatočného protokolu**, ktorý je určený špeciálne pre zabezpečenie privátnosti transakcie HTTP. Veľmi často sa používa protokol SSL (Secure Sockets Layer). Tieto sú prístupné pomocou schémy URL "https" a nie "http" vo webovom prehliadači. Verzia protokolu SSL prevzatá do dokumentov IETF sa nazýva protokol TLS (Transport Layer Security).
- Koncepcia cookies má aj svoje potenciálne problémy.
  - **Prenos citlivých informácií**. Nech napríklad používateľ používa systém internetbankingu. Používateľ sa prihlási na server, ktorý potom uloží meno konta a heslo (autentizačné údaje riadiace prístup k účtu) do cookie. Ak nie je aplikácia implementovaná starostlivo, môže byť správa obsahujúca cookie odchytená útočníkom a tento môže potom následne autentizačné údaje zneužiť. Alebo je možný iný scenár. Niekto znalý môže získať prístup do stroja používateľa a môže získať prístup do súboru, kde sú uložené cookie.
  - **Nežiadúce použitie cookies**. Teoreticky by cookie mali byť pre používateľa prínosom a nie problémom. Avšak každý server môže vytvoriť cookie z akéhokoľvek dôvodu. V niektorých prípadoch by mohol server nastaviť cookie pre účely monitorovania sídiel, ktoré používateľ navštívi. Vzhľadom k tomu, že niektoré webové prehliadače neinformujú používateľa, keď sa vytvorí cookie, používateľ si nemusí byť toho ani vedomý.

Protokol HTTP – Bezpečnosť a privátnosť

- Koncepcia cookies má aj svoje potenciálne problémy.
  - **Cookies tretej strany alebo neúmyselné cookies**. Zatiaľ čo väčšina používateľov si o cookies myslí, že cookies sú vytvorené v kontexte prostriedku, ktorý konkrétne požadujú, cookie môžu byť vytvorené ľubovoľným serverom, ktorému je zaslaná žiadosť. Napríklad pri odoslaní žiadosti http://www.firma.sk/index.htm môže táto stránka obsahovať odkaz na logo firmy, ktoré sa nachádza na serveri http://www.logafiriem.sk. Druhé sídlo môže vytvoriť cookie na počítači používateľa, aj keď používateľ nemal nikdy v úmysle k tomuto sídlu pristúpiť. Tomuto sa hovorí **cookie tretej strany**. Cookies tretích strán môžu byť použité on-line reklamnými spoločnosťami a inými na sledovanie sídiel, ktoré používateľ navštevuje. Z tohto dôvodu sú považované mnohými za nežiaduceho softvér s názvom spyware. Existuje mnoho voľne šíriteľných nástrojov, ktoré umožnia používateľovi identifikovať a odstrániť sledovacie cookie z počítača.
- Webový prehliadač **Internet Explorer umožňuje:**
  - **Umožňuje vypnúť cookies**, ale musí to urobiť používateľ.
  - **Umožňuje rozlišovať** medzi cookies prvej strany a cookies tretích strán, ale opäť si to musí zapnúť používateľ.
- Ostatné prehliadače majú sofistikovanejšie nastavenia, ktoré **umožnia používateľovi predpísať podmienky**, pri ktorých sa cookie môžu na používateľovom stroji vytvárať a pri ktorých nie.
- Niektoré prehliadače dokonca umožňujú používateľovi nastaviť niektoré lokality, od ktorých **je možné prijať cookie** a uložiť ho na používateľovom stroji **a zároveň zakazuje im prijať cookie** od ostatných.
- Lepšie prehliadače tiež dovoľujú používateľovi **vizuálnu kontrolu cookies a selektívne vymazanie** tých cookies, ktoré používateľ nechce na svojom stroji.

Ministerstvo financií
Slovenskej republiky

EFQM
Recognised for excellence
5 star - 2012

Otázky a diskusia

Ďakujem za pozornosť

KPMG
cutting through complexity™