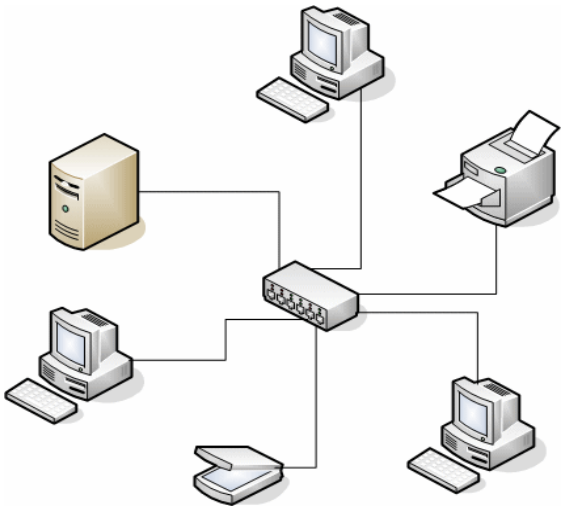
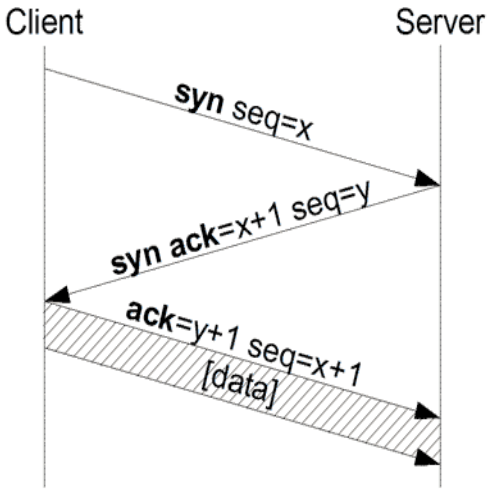
1. prednáška

158.197.31.4

56:70:B3:53:6C:EA



Internet



Čo je internet: "zariadenia a spojenia"

PC

server

wireless notebook

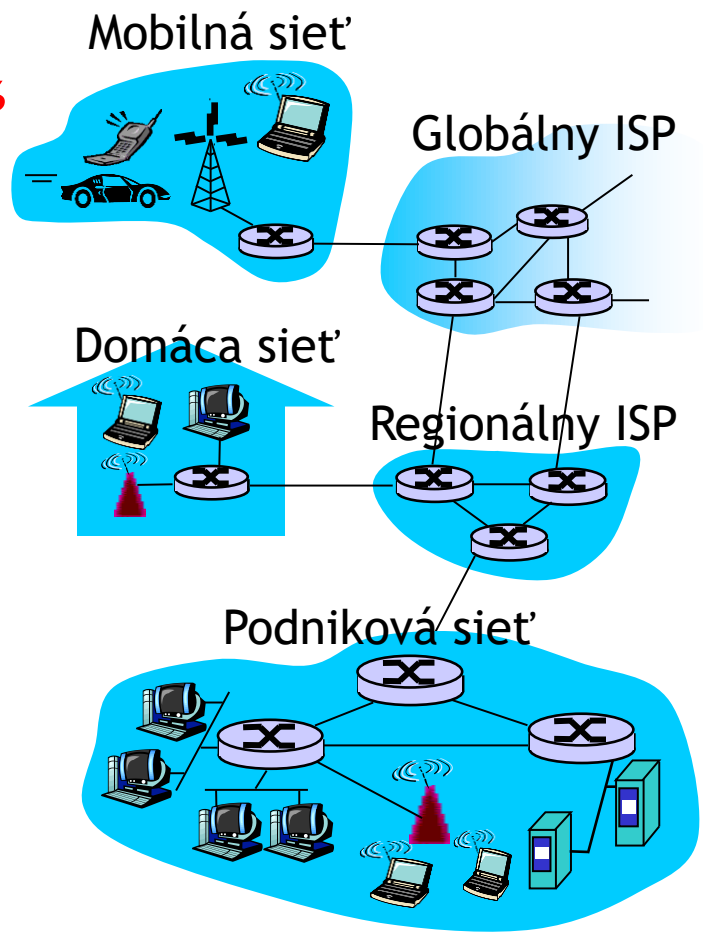
mobil

- miliardy pripojených zariadení: *hosty = koncové zariadenia*
  - spúšťajú *sieťové aplikácie*
- *spôsoby napojenia*
  - optické vlákna, káble, wifi, GPRS, 3G, LTE, satelit
  - rýchlosť spojenia závisí od *šírky pásma*
- *routre:* smerujú pakety (balíčky dát)

access pointy

spoje

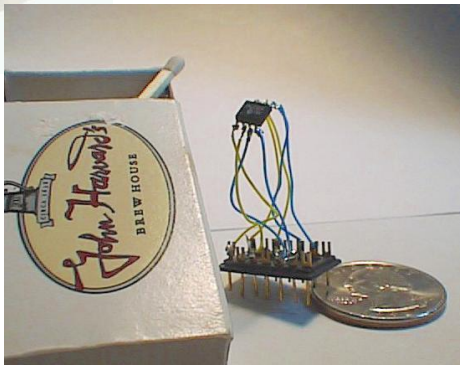
router



“Super” internetové zariadenia

Toastovač napojený na
web s predpoveďou počasia

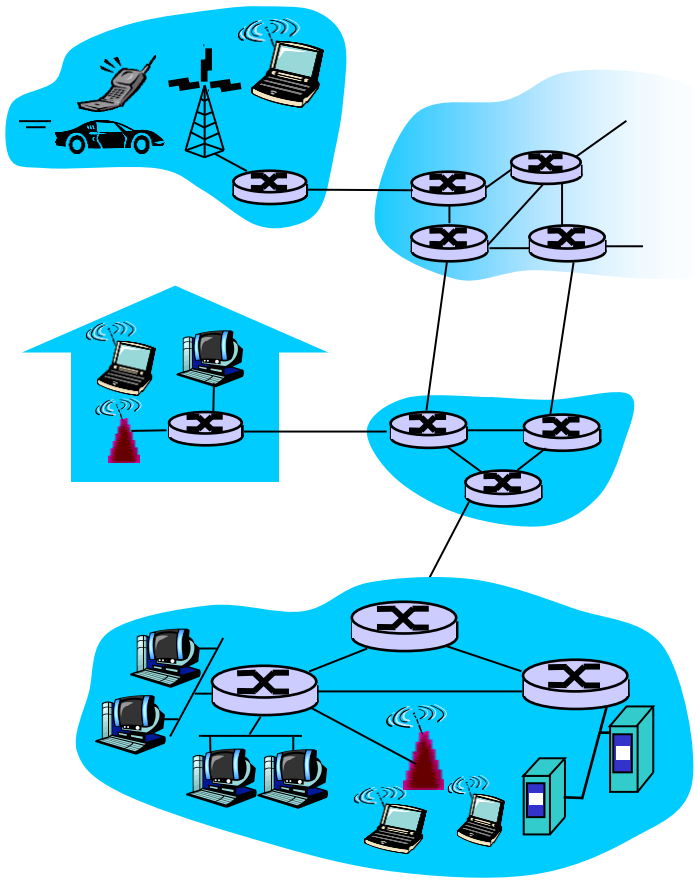
IP picture frame
http://www.ceiva.com/



Najmenší webserver na svete
http://www-ccs.cs.umass.edu/~shri/iPic.html

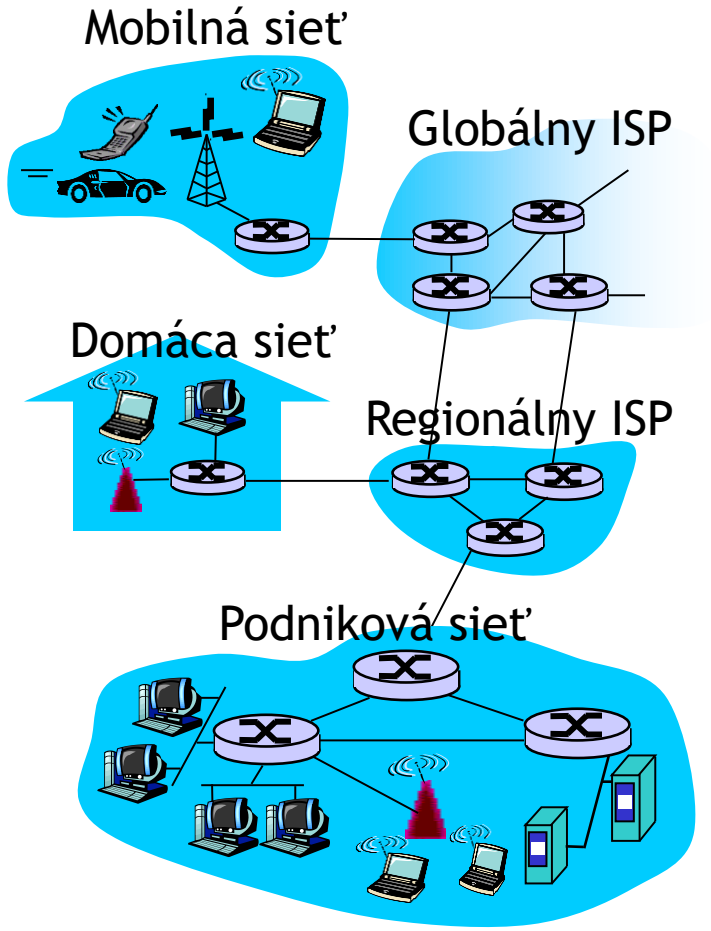
Čo je internet: množstvo služieb

- komunikácia umožňuje fungovanie distribuovaných aplikácií:
  - Web, VoIP, email, hry, e-shopy, zdieľanie súborov, sociálne siete
- komunikačné služby poskytované aplikáciami:
  - spoľahlivé doručenie dát od odosielateľa k príjemcovi
  - nespoľahlivé doručenie dát “najväčším úsilím” (“best effort”) s malou réžiou



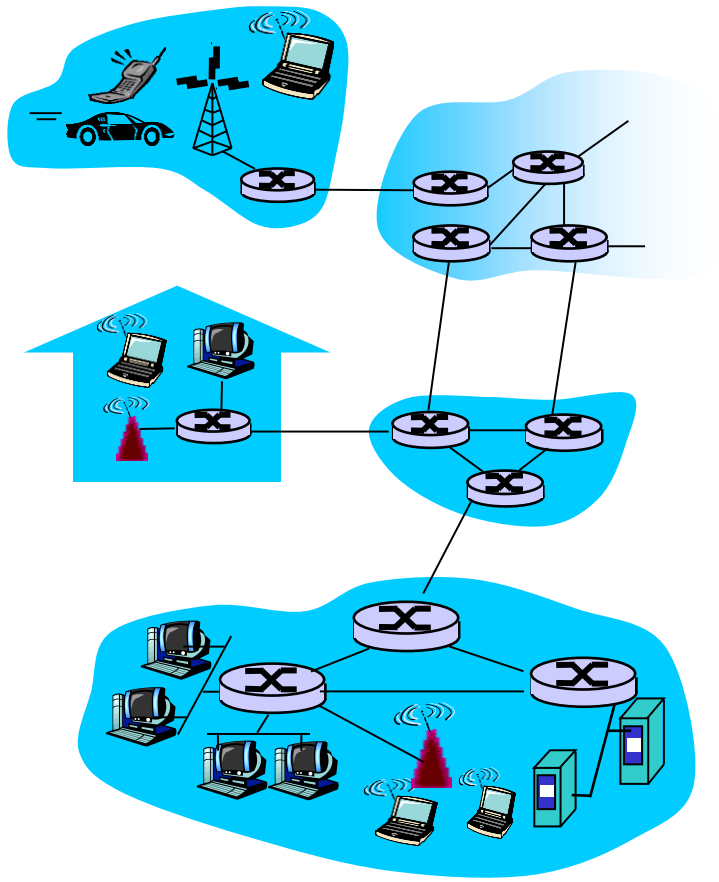
Čo je internet: "kooperácia protokolov"

- *protokoly* určujú tvar posielaných a prijímaných správ
  - napr. HTTP, XMPP, Skype
- *Internet*: "sieť sietí"
  - hierarchická štruktúra
  - verejný internet verzus súkromný intranet
- Internetové štandardy
  - RFC: Request for comments
  - IETF: Internet Engineering Task Force
  - ISO, IEEE, ITU, ...



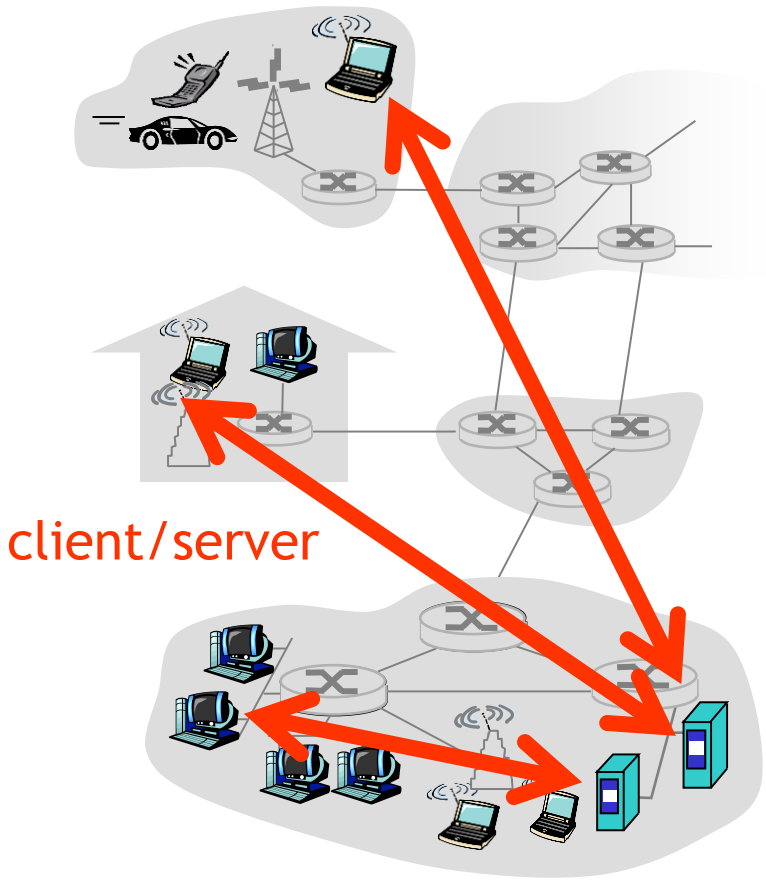
Bližší pohľad na štruktúru siete:

- "Okraje" siete:
  aplikácie a koncové zariadenia
- Prístup na sieť:
  drôtom, bezdrôtovo
- Jadro siete:
  navzájom prepojené smerovače (routre)
  - Sieť sietí
  - Iba sieťová a nižšie vrstvy



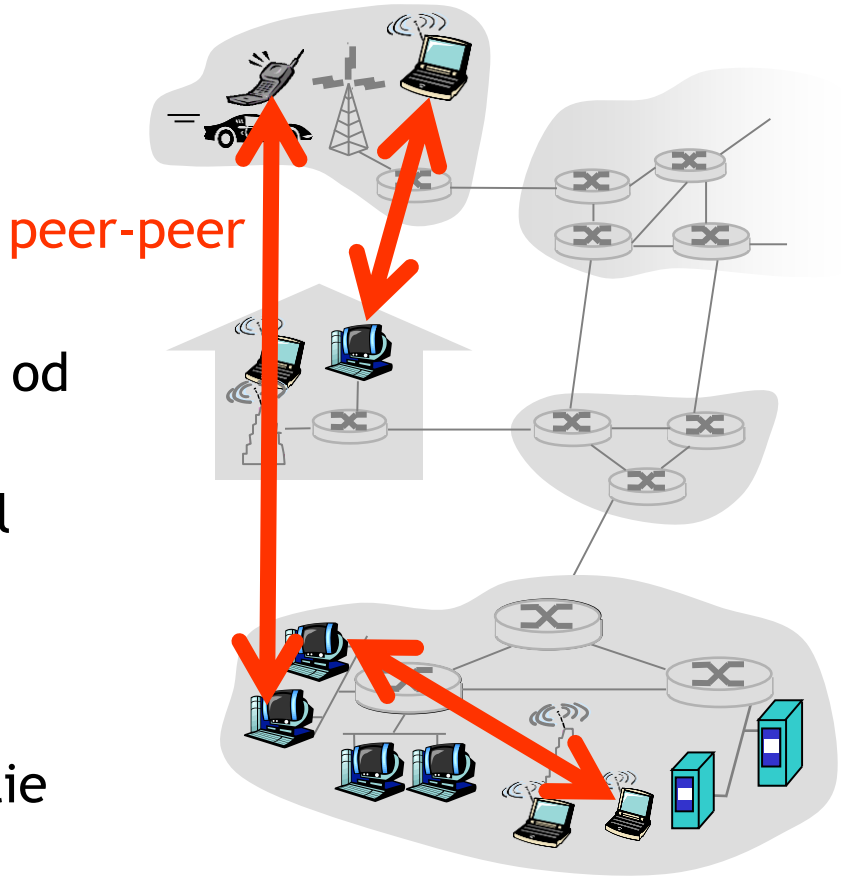
“Okraje” siete:

- koncové zariadenia:
  - spustené sieťové aplikácie
  - napr. Web, email
- klient/server model:
  - klient požaduje a prijíma službu od vždy zapnutého servera
  - napr. Web browser/server; email client/server



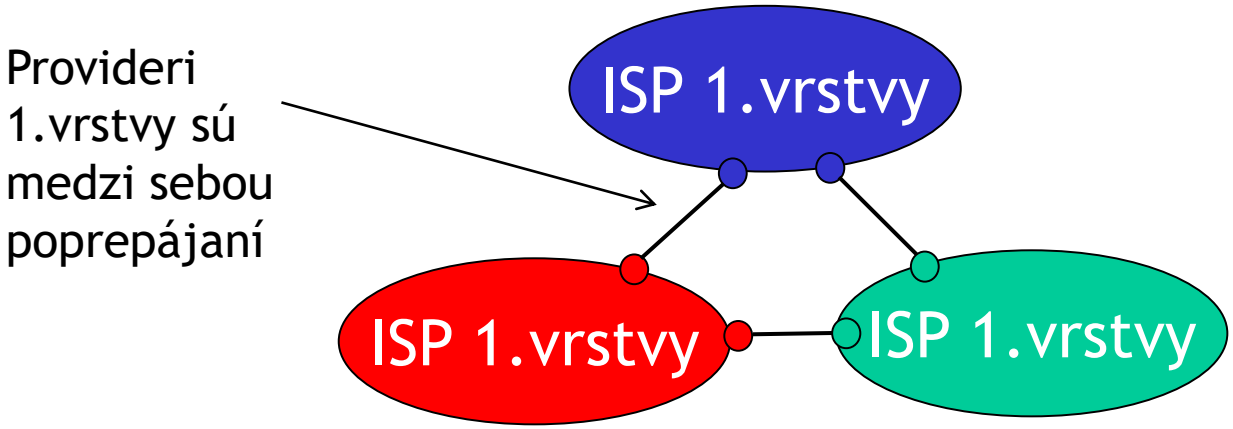
“Okraje” siete:

- koncové zariadenia:
  - spustené sieťové aplikácie
  - napr. Web, email
- klient/server model:
  - klient požaduje a prijíma službu od vždy zapnutého servra
  - napr. Web browser/server; email client/server
- peer-to-peer model:
  - minimálne (alebo žiadne) použitie “hlavných” serverov
  - napr. Skype, BitTorrent

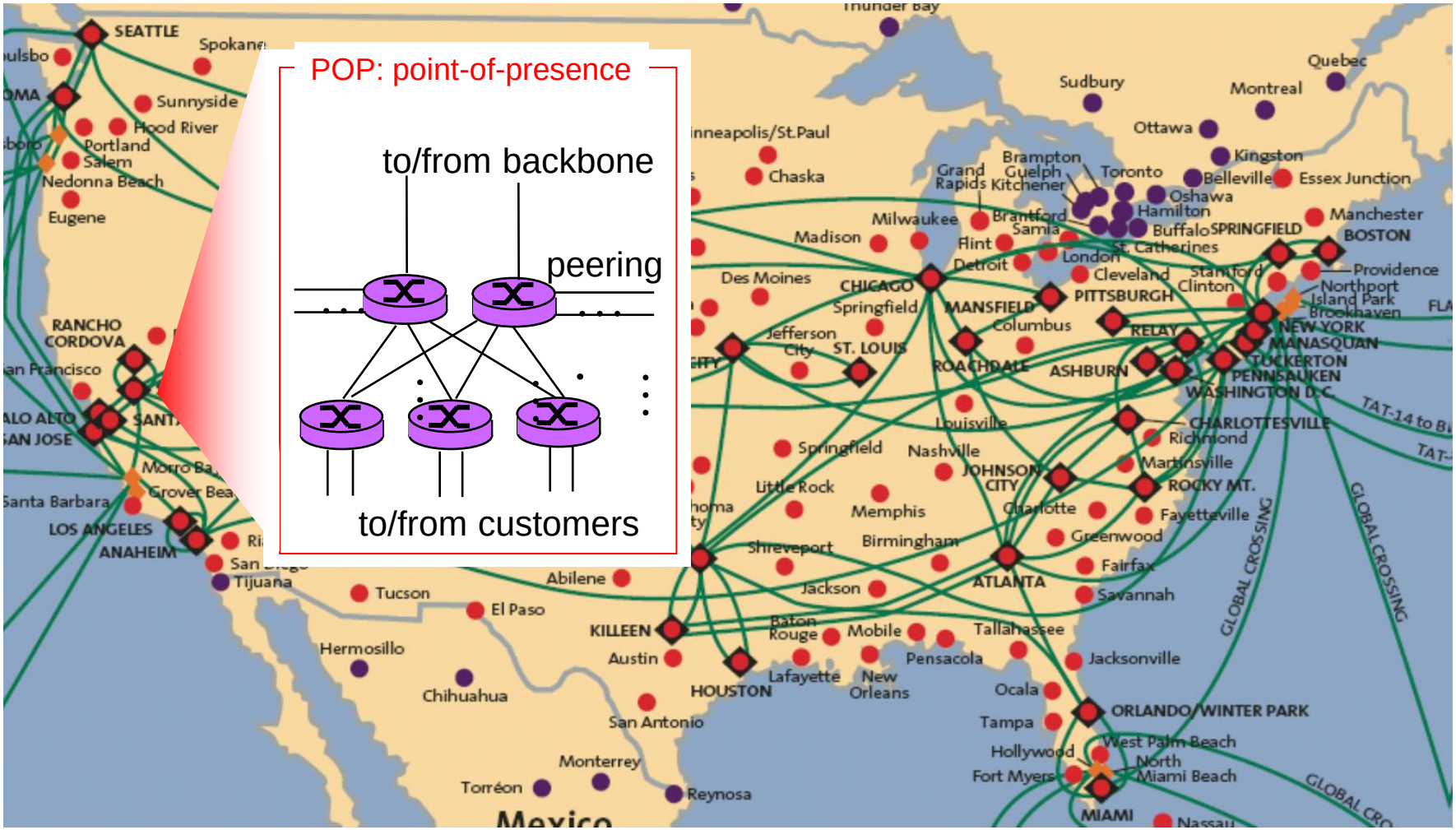


Štruktúra Internetu: sieť sietí

- Zhruba hierarchická
- V strede siete: ISPs “1.vrstvy” (e.g., Verizon, Sprint, AT&T, Cable and Wireless), národné/medzinárodné pokrytie
  - Navzájom sú si rovní

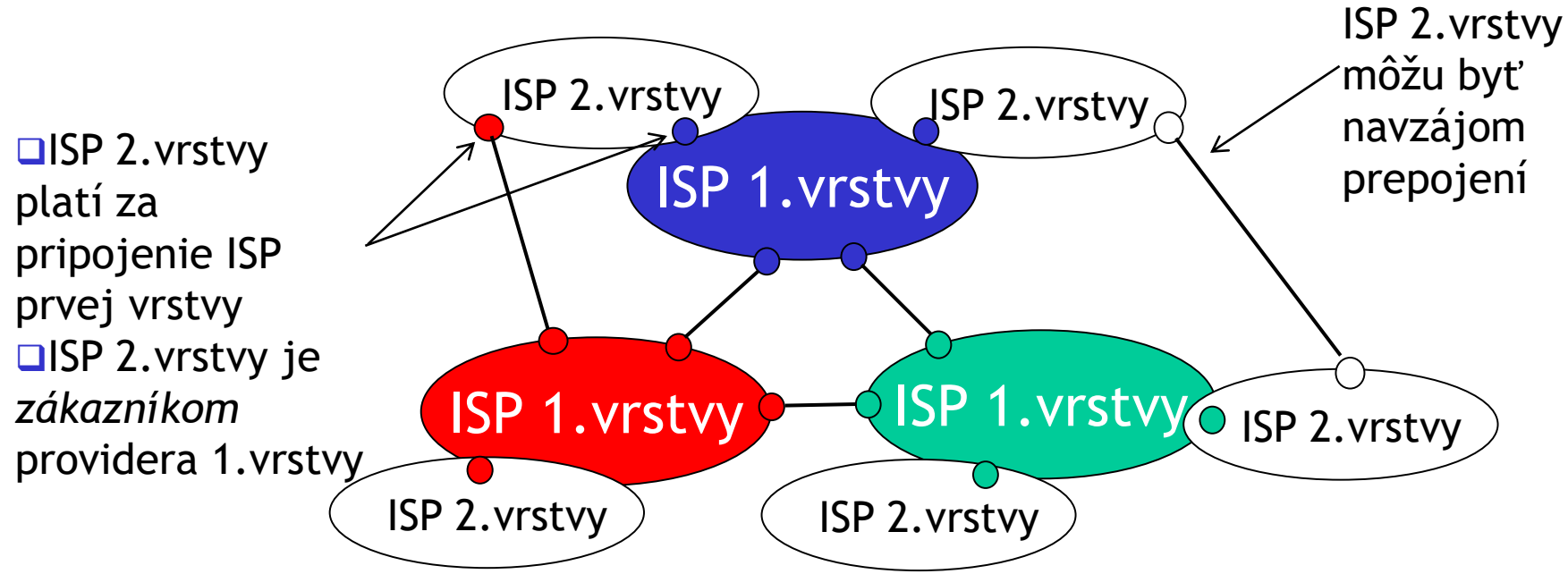


ISP 1.vrstvy: napr. Sprint



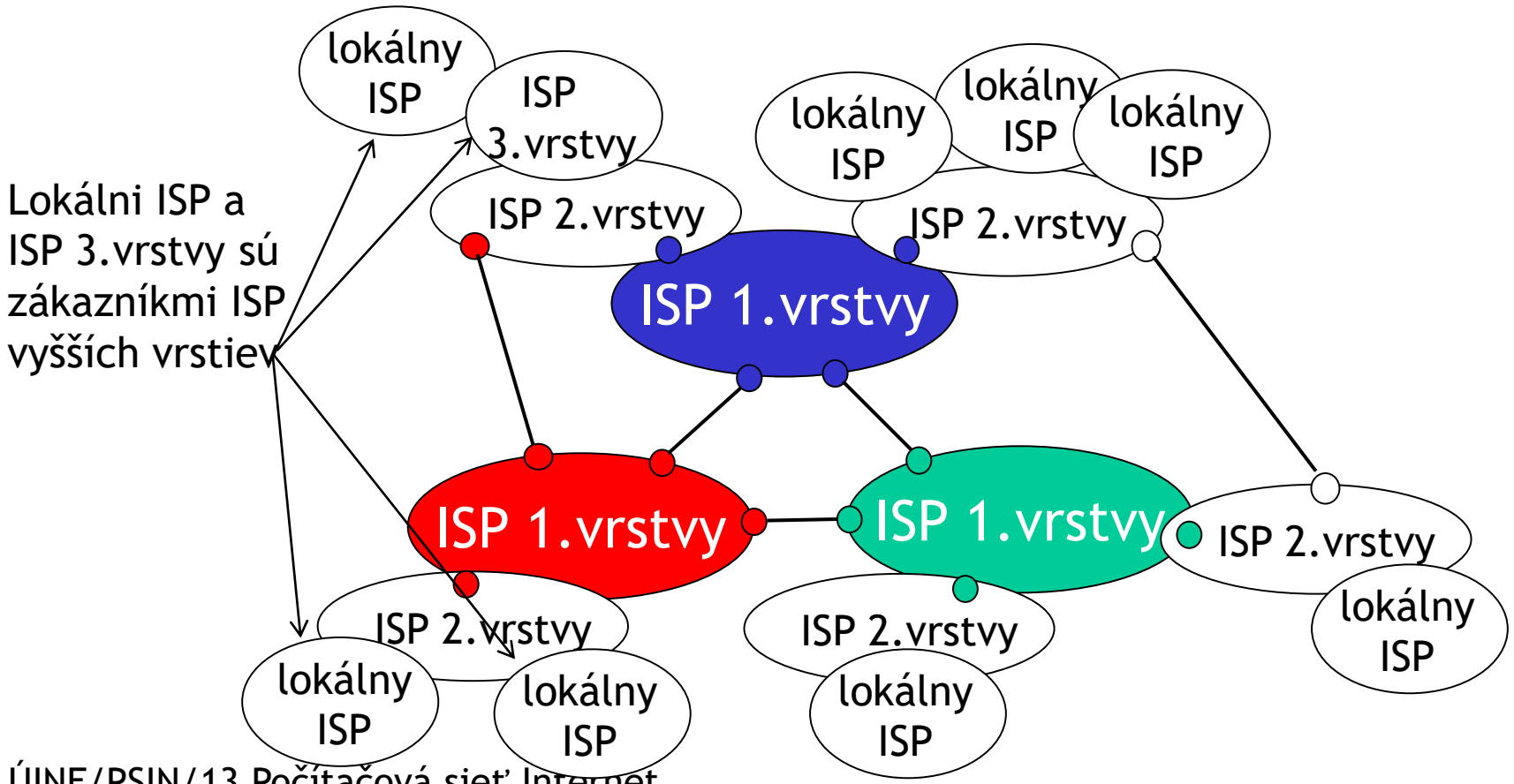
Štruktúra internetu: sieť sietí

- ISP 2. vrstvy: menší (často aj regionálni) provideri
  - Napojení na jedného alebo viac ISP 1.vrstvy, ale môžu
  - aj na iných ISP z 2.vrstvy



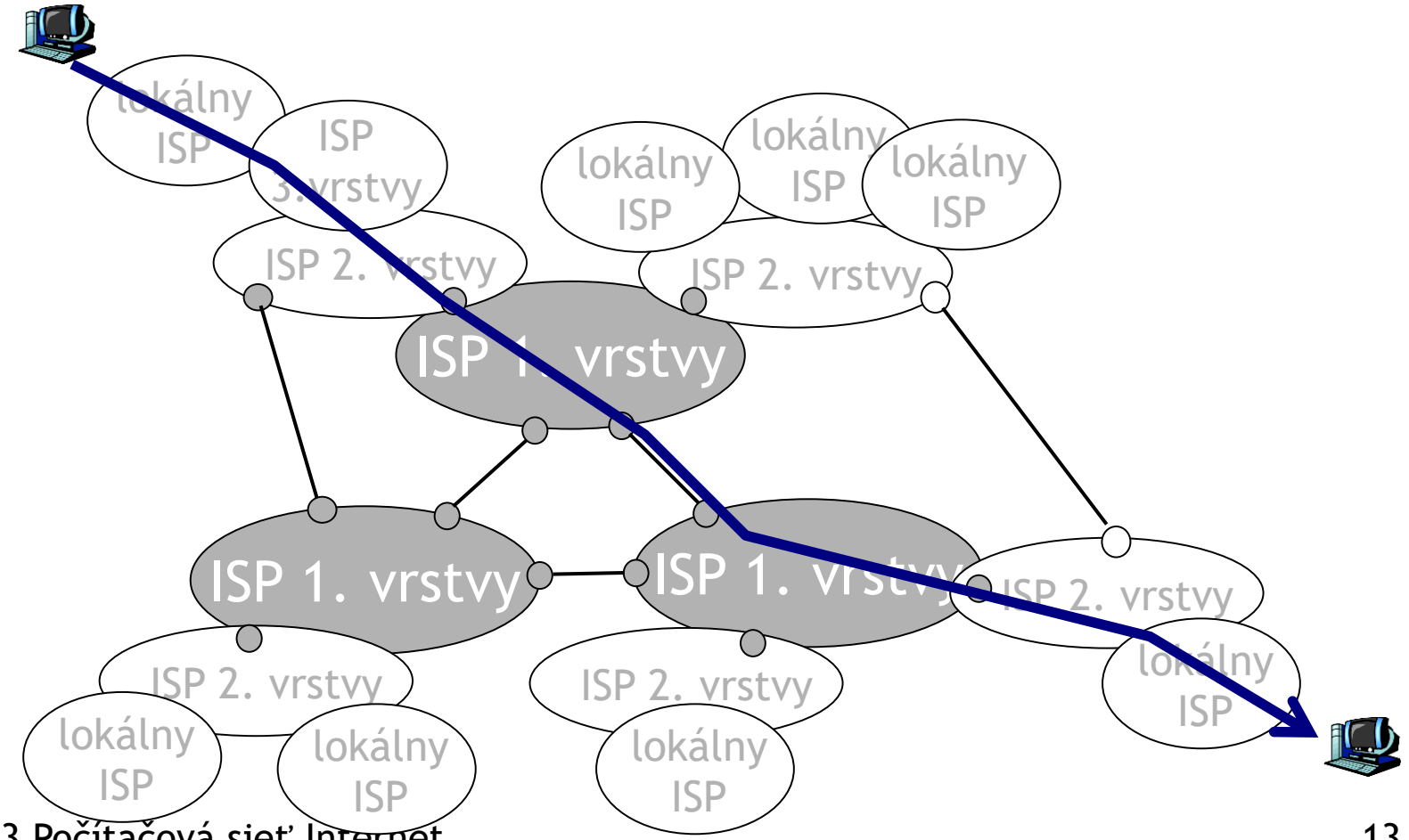
Štruktúra internetu: sieť sietí

- ISP 3.vrstvy a lokálni ISP
  - najbližšie ku koncovým zariadeniam a používateľom



Štruktúra internetu: sieť sietí

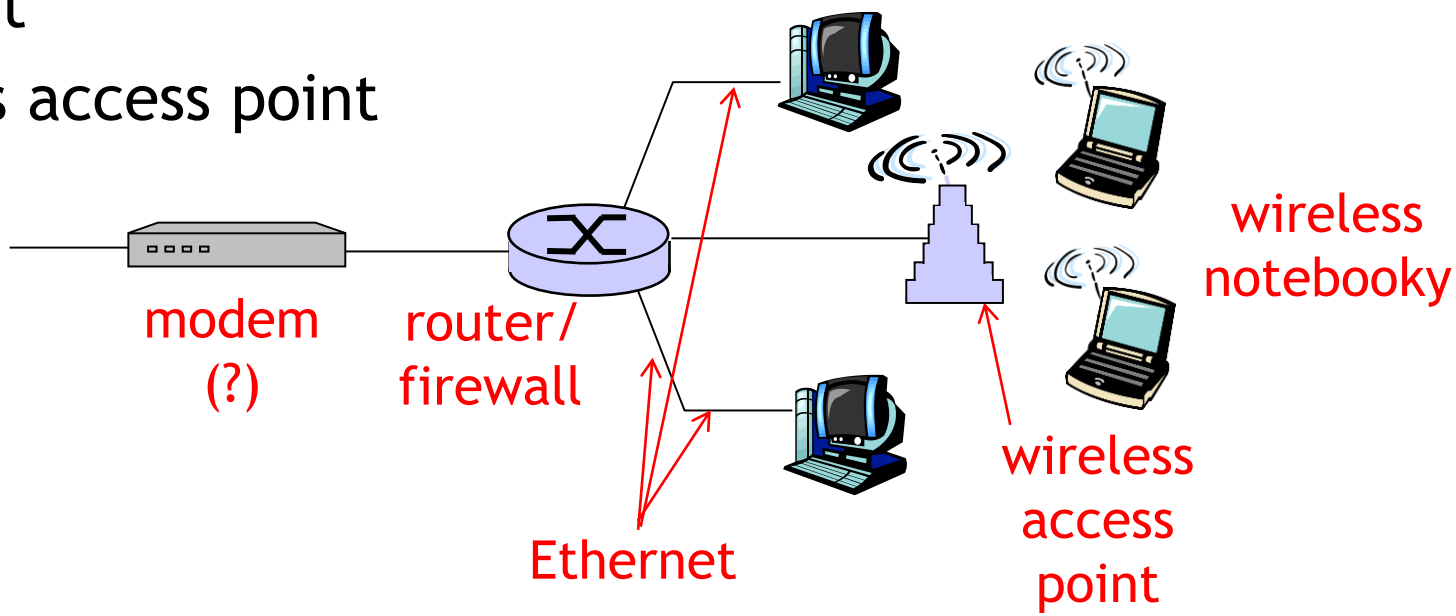
- Paket prechádza množstvom sietí



Domáce siete

Typická domáca výbava:

- DSL alebo iný modem (alebo ISP poskytuje LAN pripojenie)
- router/firewall/NAT
- ethernet
- wireless access point



Čo je to protokol?

Ľudské protokoly:

- "Koľko je hodín?"
- "Môžem sa opýtať?"
- oslovenia, pozdravy, lúčenia

… posielame vhodné "správy"

… dostávame vhodné odpovede alebo reakcie

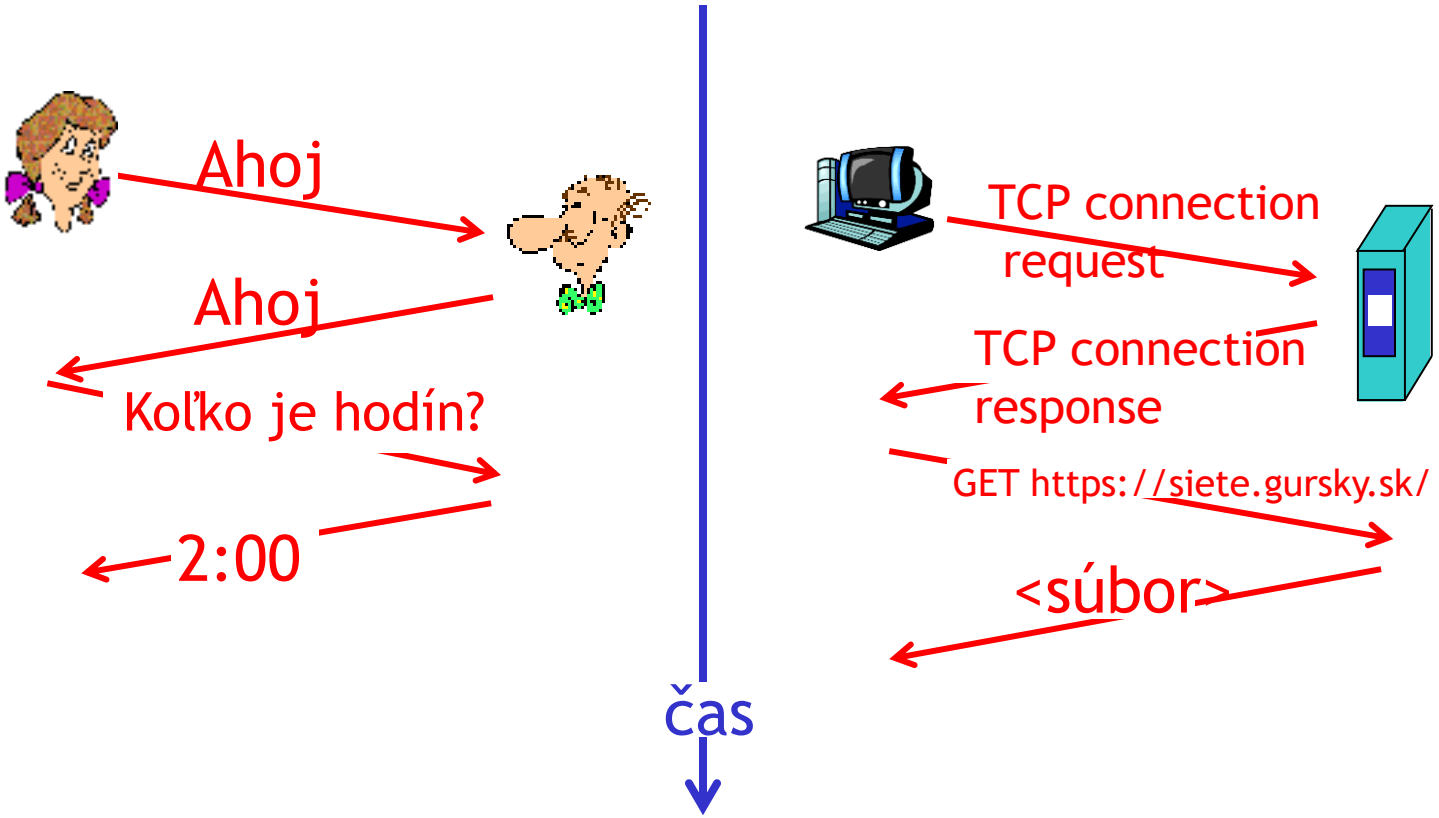
Sieťové protokoly:

- medzi procesmi a zariadeniami
- všetka komunikácia na internete je riadená protokolmi

protokoly definujú formu, poradie odoslaných a prijatých správ medzi sieťovými prvkami a akcie pri posielaní, prenášaní a prijímaní správ

Čo je to protokol?

Ľudský protokol a protokol počítačových sietí:



Vrstvy protokolov

Siete sú komplexné!

- veľa “vecí” pokope:
  - koncové zariadenia
  - routre
  - spojenia rôznymi médiami
  - aplikácie
  - protokoly
  - hardvér, softvér

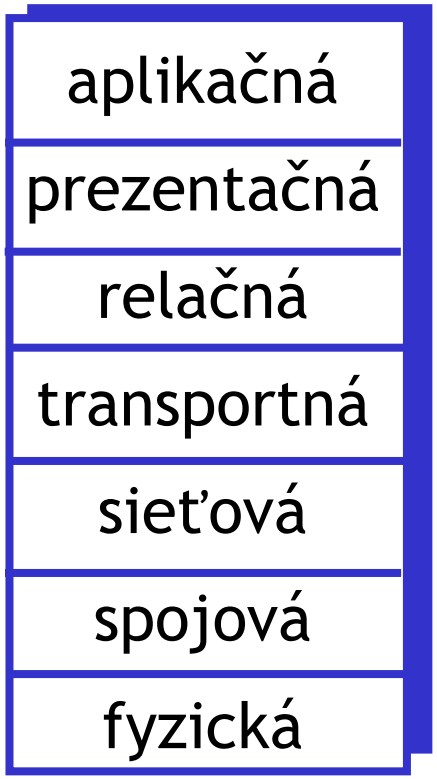
Otázka:

Existuje nejaká rozumná organizácia štruktúry sietí?

Alebo aspoň naše rozprávanie o počítačových sieťach?

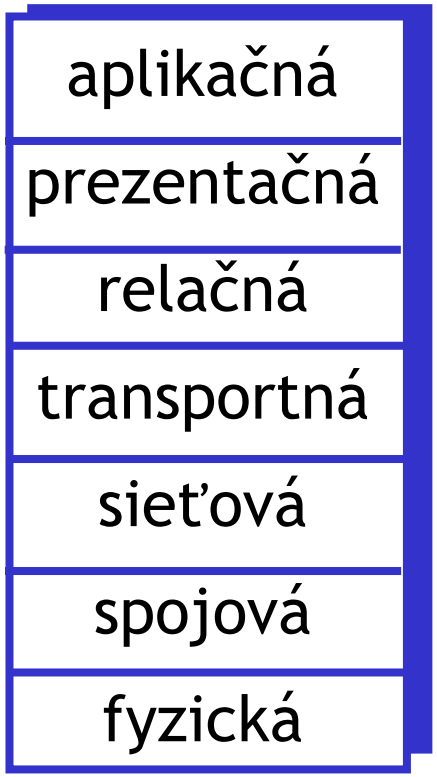
Referenčný model ISO/OSI

- transportná (transport): prenáša dáta medzi dvoma procesmi na rôznych koncových zariadeniach
- sieťová (network): smeruje pakety od odosielateľa k príjemcovi hocikde na svete
- spojová (link): prenos dát medzi danými susednými sieťovými prvkami
- fyzická (physical): prenáša fyzickým médiom jednotky a nuly

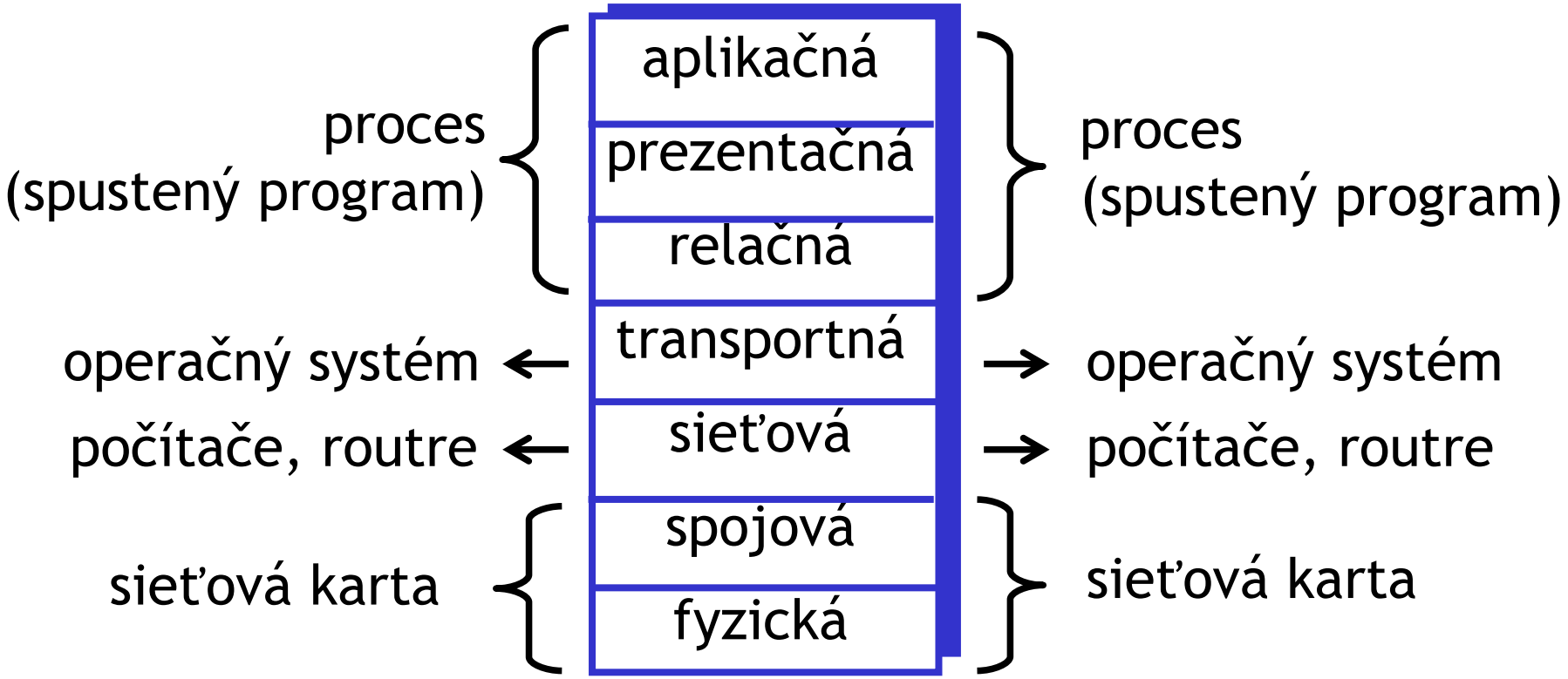


Referenčný model ISO/OSI

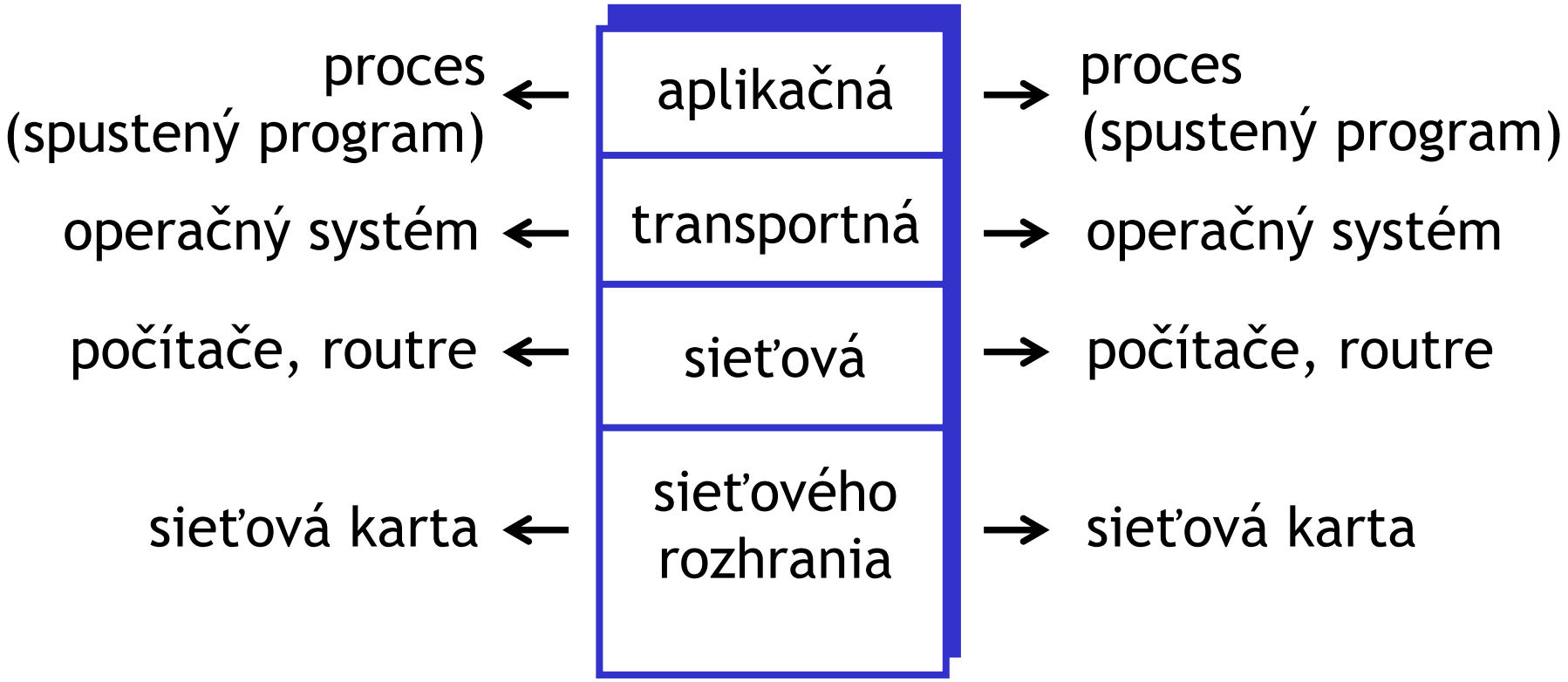
- aplikačná (application): umožňuje fungovanie sieťových aplikácií – definuje tvar a poradie správ
- prezentačná (presentation): umožňuje aplikáciám interpretovať význam dát, napr. šifrovanie, kompresia, kódovanie (znakov,..),..
- *relačná (session):* synchronizácia, kontrolné body, obnovenie relácie



Komunikujúce strany – ISO/OSI

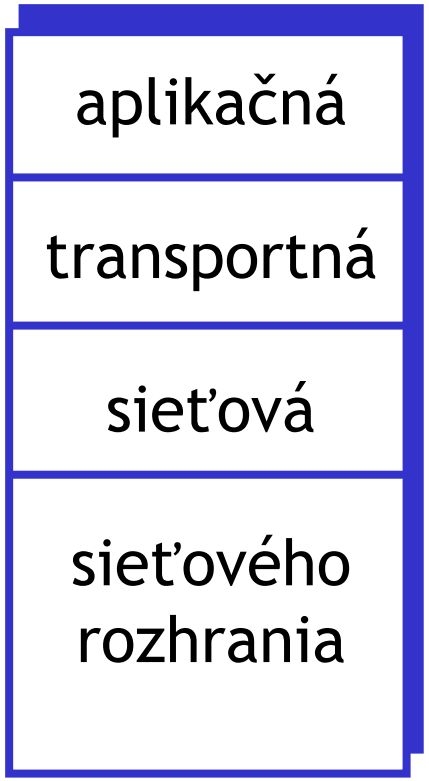


Komunikujúce strany – TCP/IP

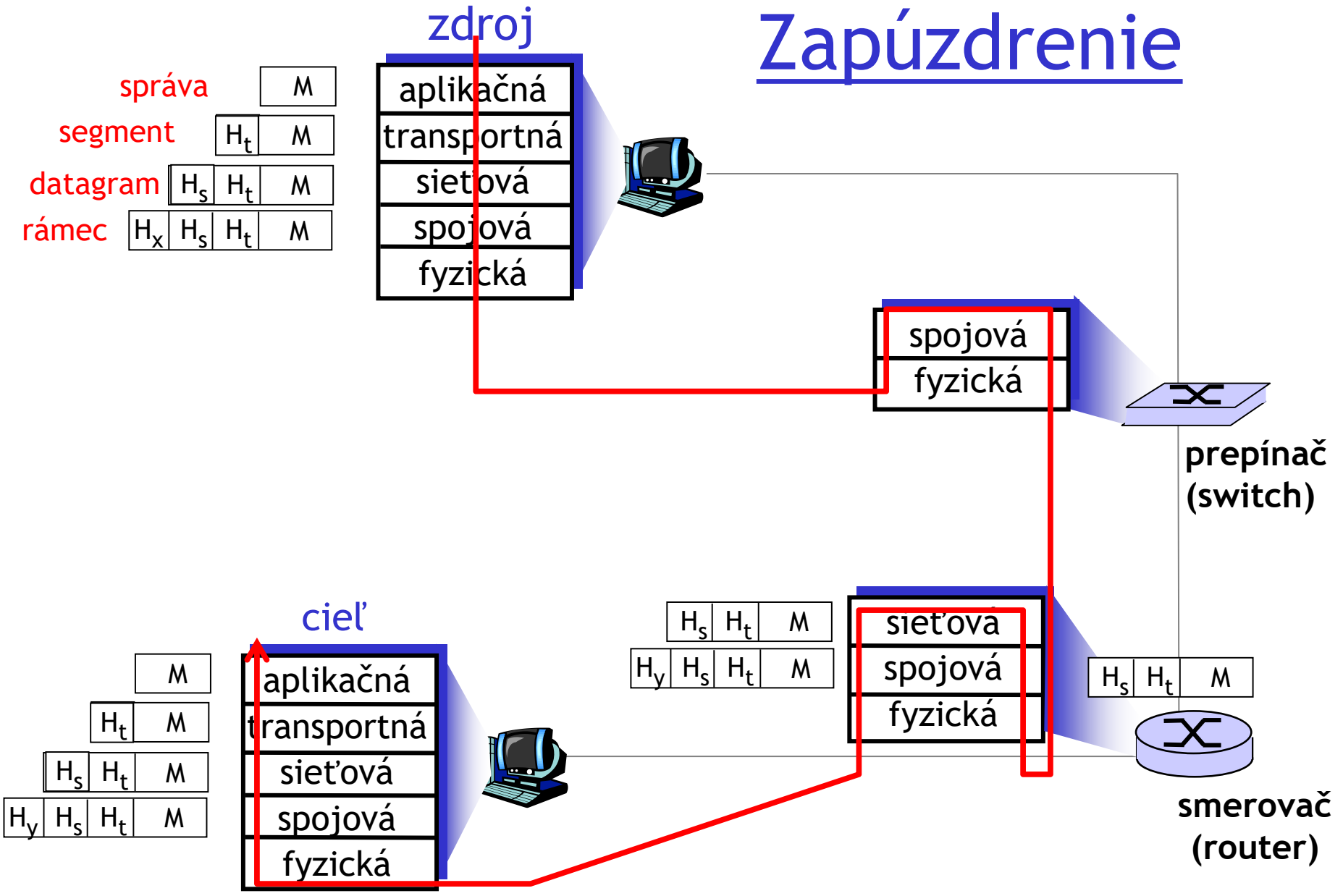


Implementácia internetu TCP/IP

- aplikačná (application): umožňuje fungovanie sieťových aplikácií – definuje tvar a poradie správ
  - prezentačná a relačná splynuli s aplikačnou
  - tieto služby musí aj tak mať implementované aplikácia, ak to potrebuje
  - a čo ak nepotrebuje?
  - HTTP, FTP, SMTP, POP, IMAP, XMPP, SSH, Torrent, ...
- transportná (transport): prenáša dáta medzi dvoma procesmi na rôznych koncových zariadeniach
  - TCP, UDP
- sieťová (network): smeruje pakety od odosielateľa k príjemcovi
  - IP, smerovacie protokoly
- sieťového rozhrania (network interface): splynutie funkcionality do technológií na prenos dát medzi susednými sieťovými prvkami a spôsobu prenášania binárnych dát
  - PPP, Ethernet



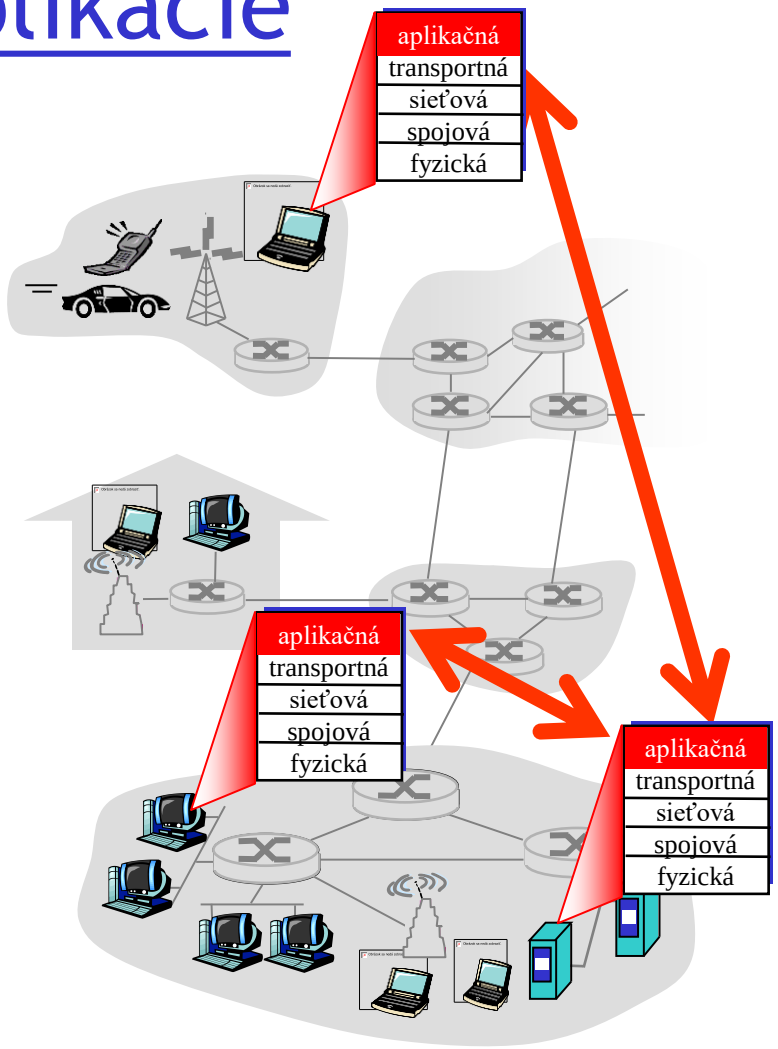
Zapúzdrenie



Vytváranie sieťovej aplikácie

programy

- bežia na (rôznych) *koncových systémoch*
- komunikujú prostredníctvom siete
- napr. web server komunikuje s browserom
- zariadenia jadra siete nespúšťajú používateľské aplikácie



Komunikácia procesov

Proces: program bežiaci na počítači

- V rámci jedného počítača procesy obvykle komunikujú medziprocesovou komunikáciou (definovanou v OS).
- procesy na rôznych počítačoch komunikujú prostredníctvom správ

Klientský proces: proces ktorý inicializuje komunikáciu

Serverový proces: proces, ktorý čaká na to, že sa naňho niekto napojí

- Poznámka: aj aplikácie vo všetkých P2P architektúrach majú klientské a serverovské procesy

Protokol aplikačnej vrstvy definuje:

- Typy správ, ktoré sa vymieňajú,
  - Napr. požiadavky, odpovede
- Syntax správ:
  - Z čoho sa skladajú jednotlivé správy, rozsahy hodnôt
- Sémantika správ
  - Význam správ a informácií v nich
- Pravidlá pre to, kedy, za akých okolností, a ako si budú procesy posielať správy

Verejné protokoly:

- Definované v RFC
- Umožňujú interoperabilitu
- napr. HTTP, SMTP

Proprietárne protokoly:

- napr. Skype

Adresácia procesov (sieťová vrstva)

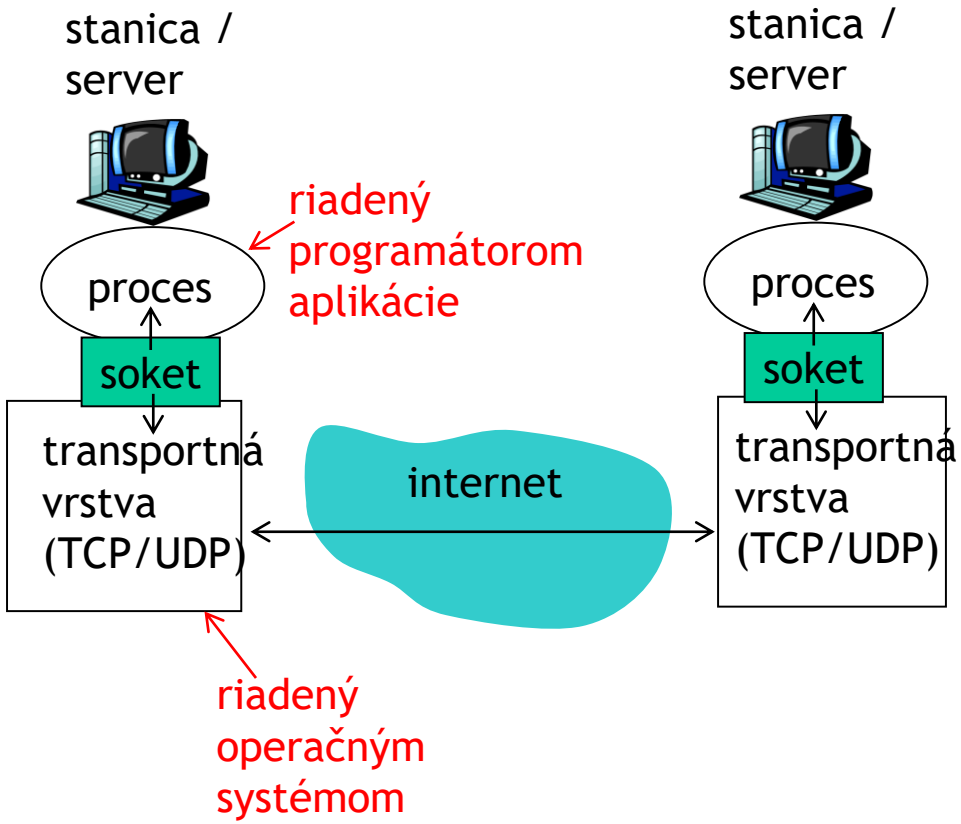
- aby sa procesu dali posielať správy, musíme ho na internete presne identifikovať
- počítač musí mať jedinečnú IP adresu
- stačí IP adresa počítača na identifikáciu procesu sieťovej aplikácie?

Adresácia procesov (sieťová a transportná vrstva)

- aby sa procesu dali posielať správy, musíme ho na internete presne identifikovať
- počítač musí mať jedinečnú IP adresu
- stačí IP adresa počítača na identifikáciu procesu sieťovej aplikácie?
  - Nie! Na jednom počítači môže byť viac sieťových procesov
- *identifikátor* zahŕňa IP adresu a číslo portu(portov) priradené k procesu na počítači.
- Príklady čísiel portov:
  - HTTP server: 80
  - Mailový server: 25
- Na poslanie HTTP správy na web server web.ics.upjs.sk potrebujeme:
  - IP adresu: 158.197.31.29
  - Číslo portu: 80

Komunikácia s transportnou vrstvou - Sokety

- procesy posielajú a prijímajú správy prostredníctvom svojich soketov
- Čo je soket?
  - rozhranie, cez ktoré komunikuje sieťová aplikácia s implementáciou protokolu transportnej vrstvy v operačnom systéme



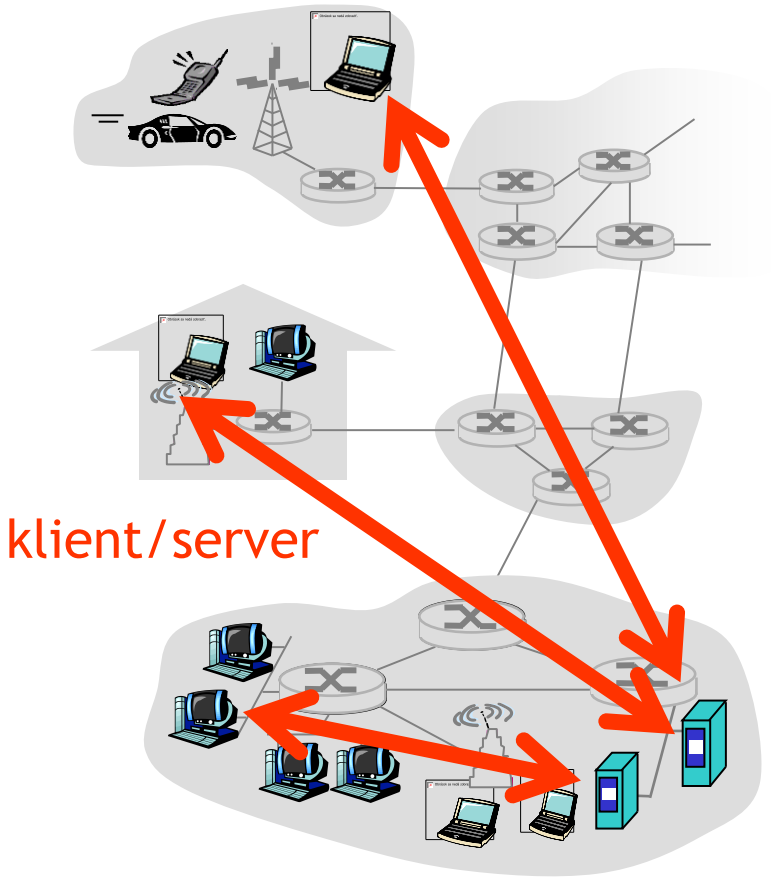
API: (1) iné pre rôzne transportné protokoly

(2) umožňuje nastaviť niektoré parametre nižších vrstiev

Architektúry sieťových aplikácií

- Klient-server
- Peer-to-peer (P2P)
- Hybrid oboch

Architektúra klient-server



server:

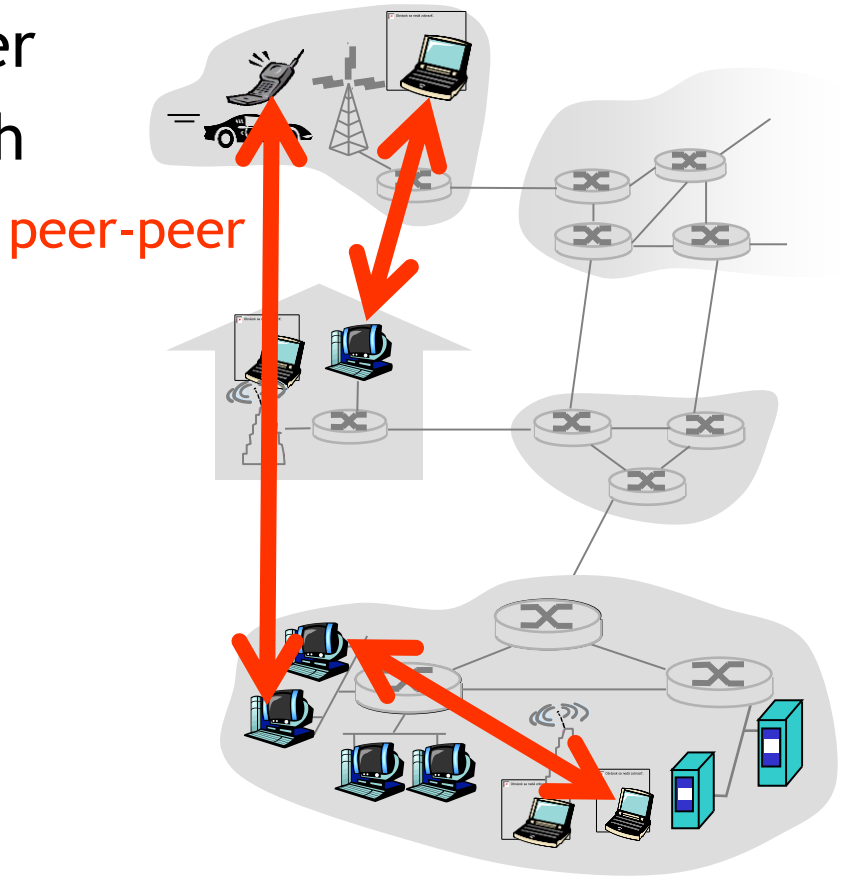
- stále zapnutý počítač
- pevná IP adresa
- serverové farmy pre lepšiu škálovateľnosť

klienti:

- komunikujú so serverom
- môžu sa priebežne odpájať
- môžu mať dynamickú IP adresu
- nekomunikujú medzi sebou

Čistá P2P architektúra

- *nemá* stále zapnutý server
- na ľubovoľných koncových systémoch
- komunikujú medzi sebou
- peerovia sú prerušovane napojení a môžu meniť IP adresy
- príklad: Gnutella



Vysoko škálovateľné, ale ťažko manažovateľné

Hybrid Klient-servera a P2P

Skype

- program na telefonovanie cez IP
- centrálny server: hľadá adresu cieľového používateľa cez klient-server
- komunikácia už typu peer-peer: priame spojenie (nie cez server)

Instant messaging

- chat medzi dvoma používateľmi ako P2P
- centralizovane sa zisťuje pripojenie klientov
- klienti si v centrálnom serveri registrujú svoje IP adresy, keď sa prihlásia
- klienti kontaktujú server, aby zistili IP adresy kontaktov, s ktorými chcú rozprávať

Web a HTTP

označenia

- Webová stránka sa skladá z objektov
- Objekt môže byť HTML súbor, obrázok, Java applet, audio súbor, CSS súbor…
- Web stránku tvorí základný HTML súbor, ktorý obsahuje niekoľko odkazov na ďalšie objekty
- Adresa objektu je URL
- Príklad URL:

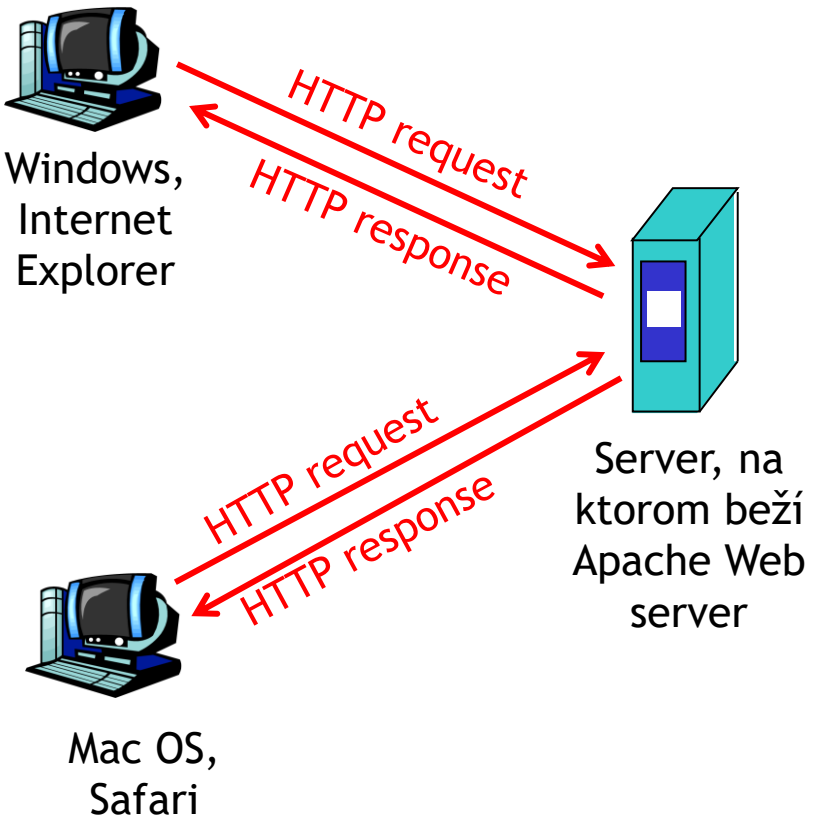
`www.inštitúcia.sk/oddelenie/obr.gif`

doménové meno — cesta

Náhľad na HTTP

HTTP: hypertext transfer protocol

- Protokol aplikačnej vrstvy pre Web
- klient/server model
  - *klient:* prehliadač, ktorý žiada, dostáva a zobrazuje webové objekty
  - *server:* webový server odosiela objekty v odpovediach na požiadavky
- HTTP 1.0: RFC 1945 (1996)
- HTTP 1.1: RFC 2068 (1997)
- HTTP 2.0: RFC 7540 (2015)



Náhľad na HTTP

Používa TCP:

- klient inicializuje TCP spojenie (vytvorí soket) so serverom na porte 80
- server akceptuje TCP spojenie od klienta
- browser (HTTP klient) a web server (HTTP server) si posielajú HTTP správy (správy aplikačného protokolu)
- TCP spojenie sa zavrie

HTTP je "bezstavový"

- server si neuchováva históriu predchádzajúcich požiadaviek klienta

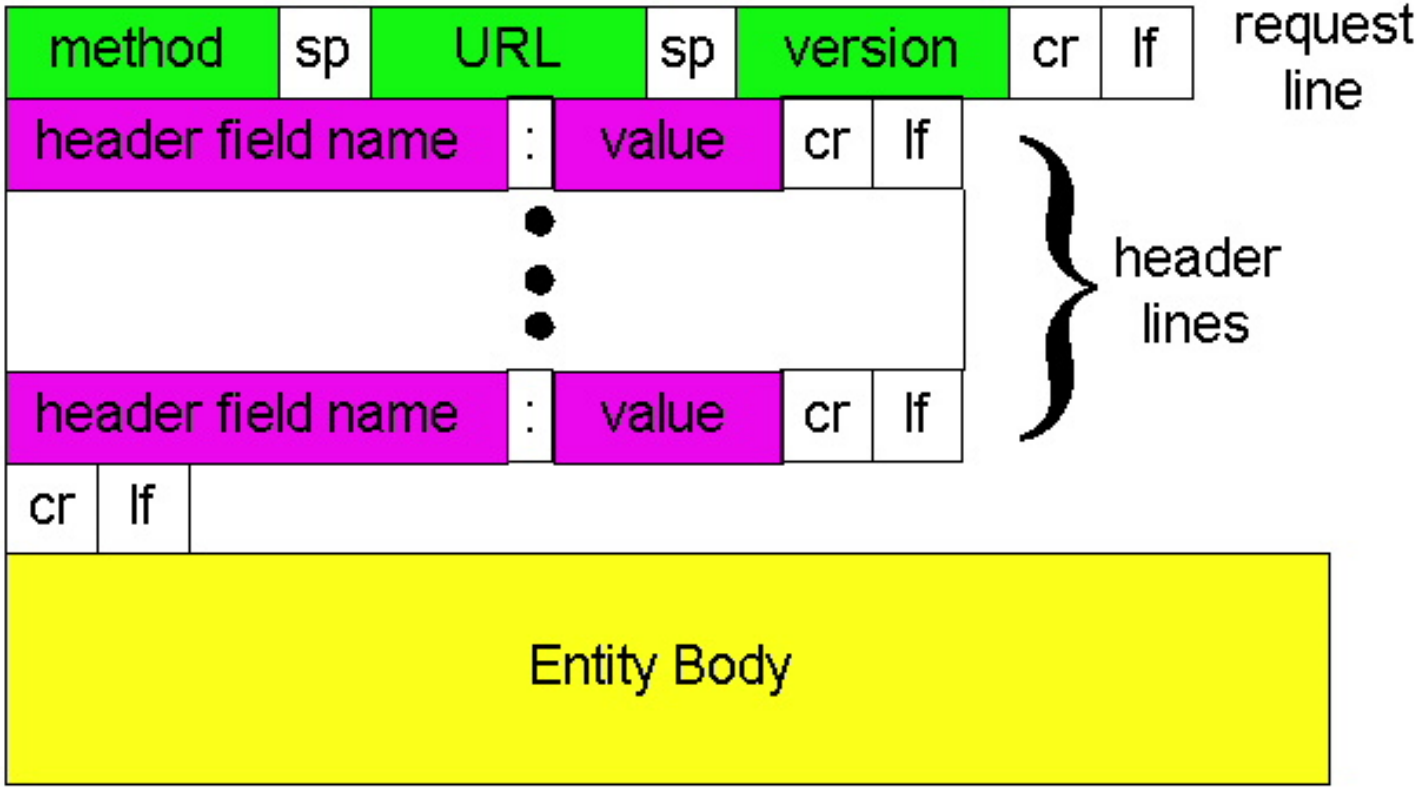
Stavové protokoly sú zložité!

- musí sa spravovať história
- po zlyhaní klienta alebo servera sa ich pohľad na to, čo je to posledný stav, môže líšiť a musia sa synchronizovať

HTTP/1.1 požiadavka

- Dva typy HTTP správ: *požiadavka(request)* a *odpoveď(response)*
- HTTP/1.1 požiadavka:
  - ASCII (ľudsky čitateľný tvar)

HTTP/1.1 požiadavka všeobecne



Odosielanie formulárových dát

POST:

- Pri odosielaní formulárov z webstránok lepšia alternatíva ku GET
- Dáta sú posielané na server v “entity body”

Spôsob cez URL:

- Používa príkaz GET
- Dáta sú súčasťou URL adresy:

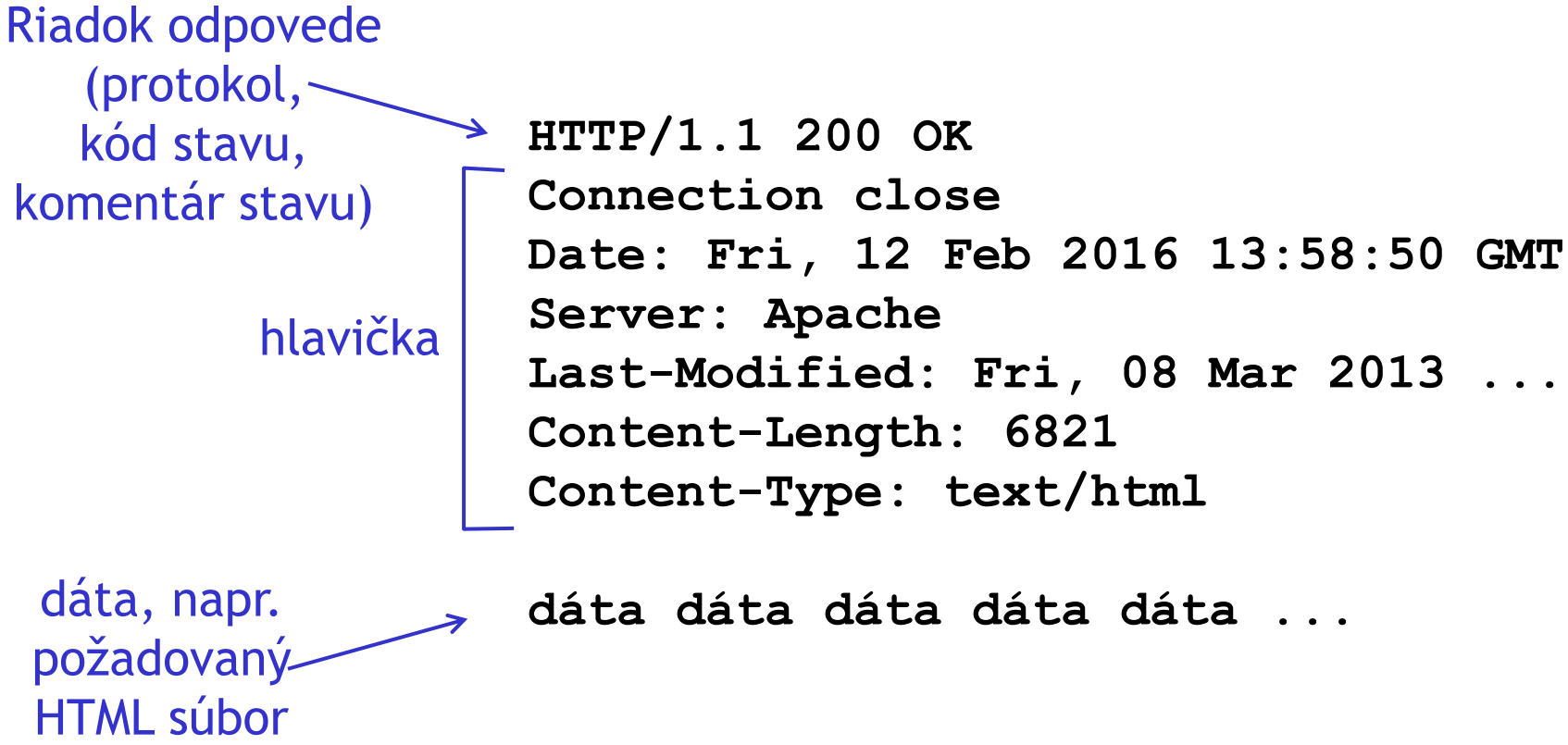
```
www.somesite.com/animalsearch?monkeys&banana
```

Príkazy požiadaviek

- GET, POST
- HEAD
  - "neposielaj žiadaný objekt, iba hlavičku"
- PUT
  - posiela súbor v "entity body" na objekt špecifikovaný v URL
- DELETE
  - vymaže objekt na adrese špecifikovanej v URL

Využitie v CRUD protokoloch napr. REST

HTTP/1.1 odpoveď



Stavy HTTP odpovede

Niektoré príklady prvého riadka odpovede servera:

200 OK

- požiadavka úspešná, požadovaný objekt nasleduje pod hlavičkou

301 Moved Permanently

- požadovaný objekt je presunutý, nová pozícia je špecifikovaná v hlavičke v časti “Location:”

400 Bad Request

- server nerozumie požiadavke

404 Not Found

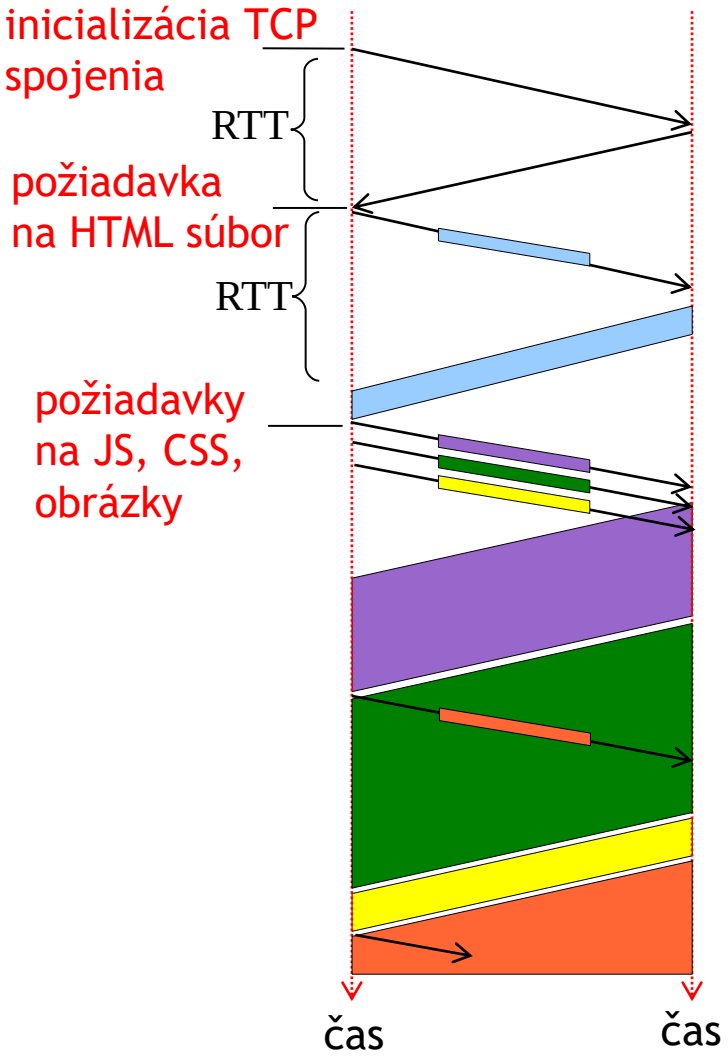
- požadovaný objekt sa nenašiel

505 HTTP Version Not Supported

- server nepodporuje požadovanú verziu HTTP protokolu

HTTP/1.1 spojenia

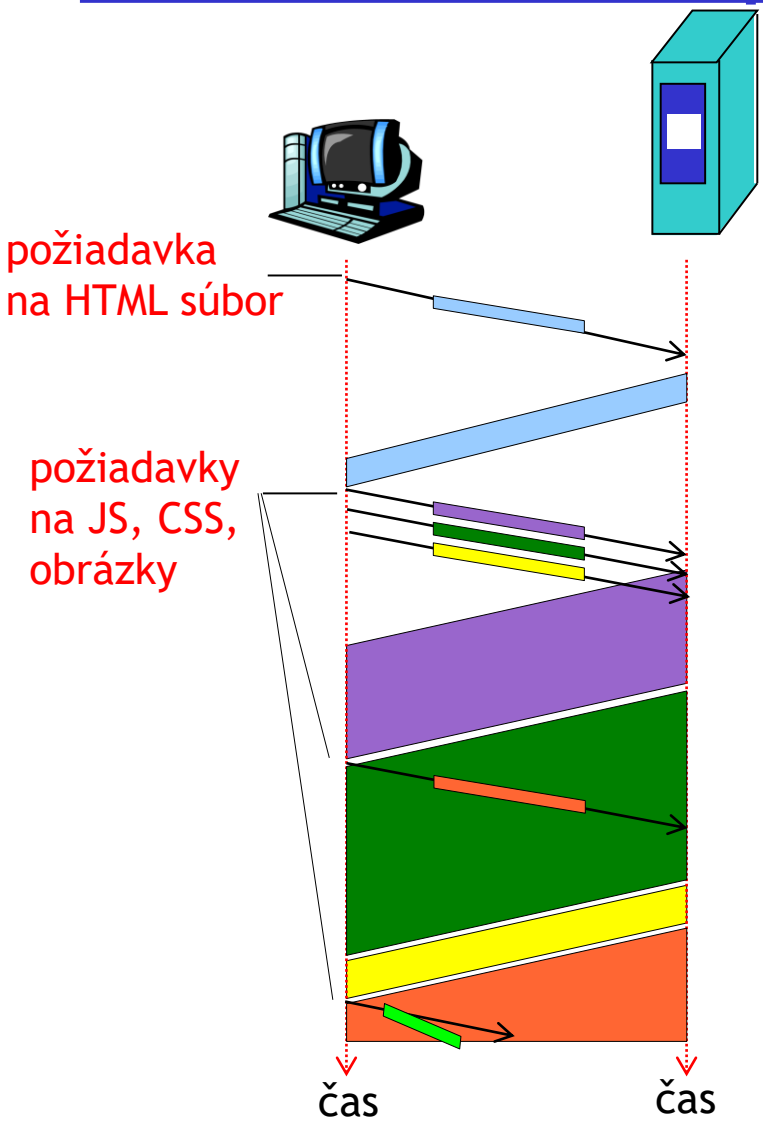
- Perzistentné spojenie
  - Nechám spojenie otvorené, pokiaľ nenatiahnem všetky objekty.
- Pipelinig (reťazenie požiadaviek a odpovedí)
  - Po jednej požiadavke nečakám na odpoveď, ale zašlem všetky požiadavky za sebou.

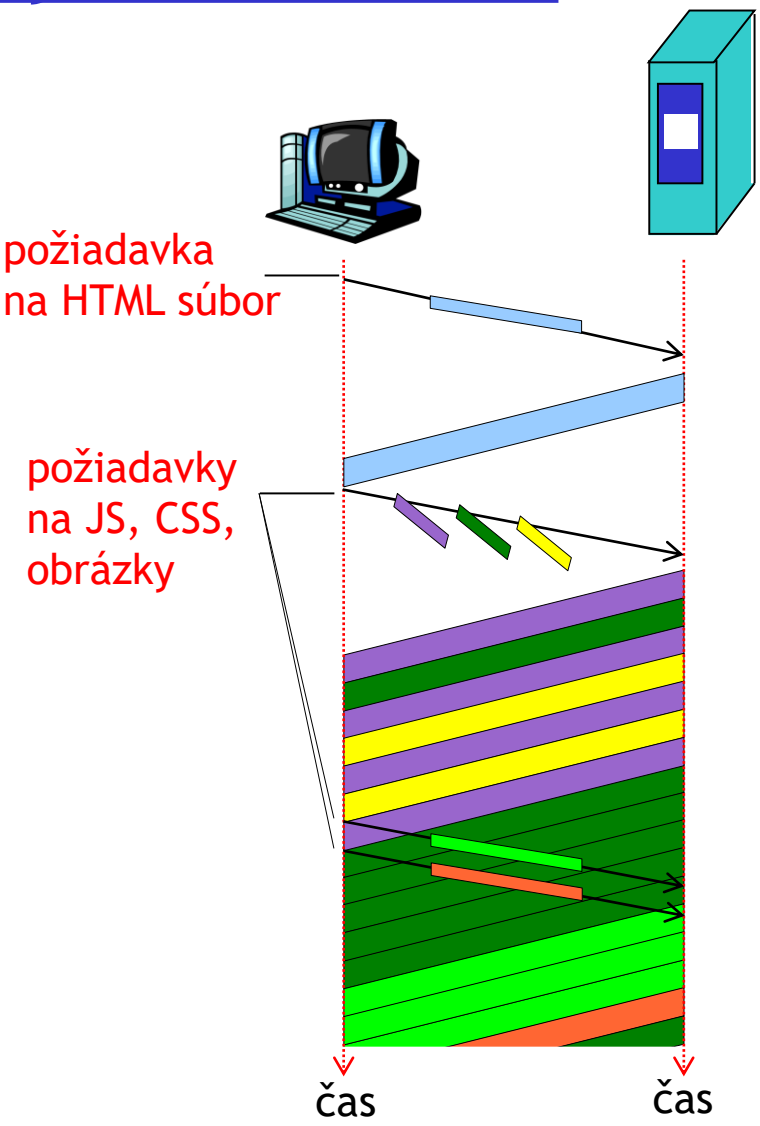


HTTP/2 – od mája 2015

- používanie binárnych rámcov namiesto pôvodných textových HTTP požiadaviek a odpovedí (všetky majú rovnaký tvar 9 bajtovej hlavičky)
- 10 typov rámcov (SETTINGS, HEADERS, DATA, CONTINUATION,...)
- HEADERS rámec požiadavky, ako náprotivok hlavičky HTTP/1.x požiadavky, definuje nový prúd dát v rámci spojenia, ktorý má zabezpečiť stiahnutie jedného objektu. Všetky prúdy sú posielané cez to isté spojenie.
- každý prúd dát môže mať rôznu prioritu, ktorá umožní zasielanie rámcov jedného objektu častejšie ako iného
- nastavovanie vzájomných závislostí prúdov (ak mi nepošleš tento objekt, tak objekty na ňom závislé tiež neposielaj)
- kompresia hlavičiek, kde ďalšie hlavičky obsahujú iba rozdiely oproti predchádzajúcim hlavičkám
- možnosť pre server zasielať objekty bez vyžiadania, ak server predpokladá, že prehliadač bude daný objekt potrebovať
- možnosť použitia spojenia so serverom pre viaceré domény, ak sú hostované na serveri s rovnakou IP adresou

HTTP/1.1 vs. prúdy v HTTP/2





HTTP/2: Kompresia hlavičiek

Chcem poslať:

:method	GET
:scheme	https
:host	siete.gursky.sk
:path	/obr.jpg
user-agent	Mozilla/5.0...
content-type	image/jpg

Predchádzajúce:

:method	GET
:scheme	https
:host	siete.gursky.sk
:path	/head.png
user-agent	Mozilla/5.0...
content-type	image/png
connection	Keep-alive
cookie	id=d38rb8x39...

Posielam:

:path	/obr.jpg
content-type	image/jpg

Posielané atribúty hlavičky sú v HPACK formáte používajúcom kompresiu cez Huffmanovo kódovanie

Stav používateľovej komunikácie: cookies

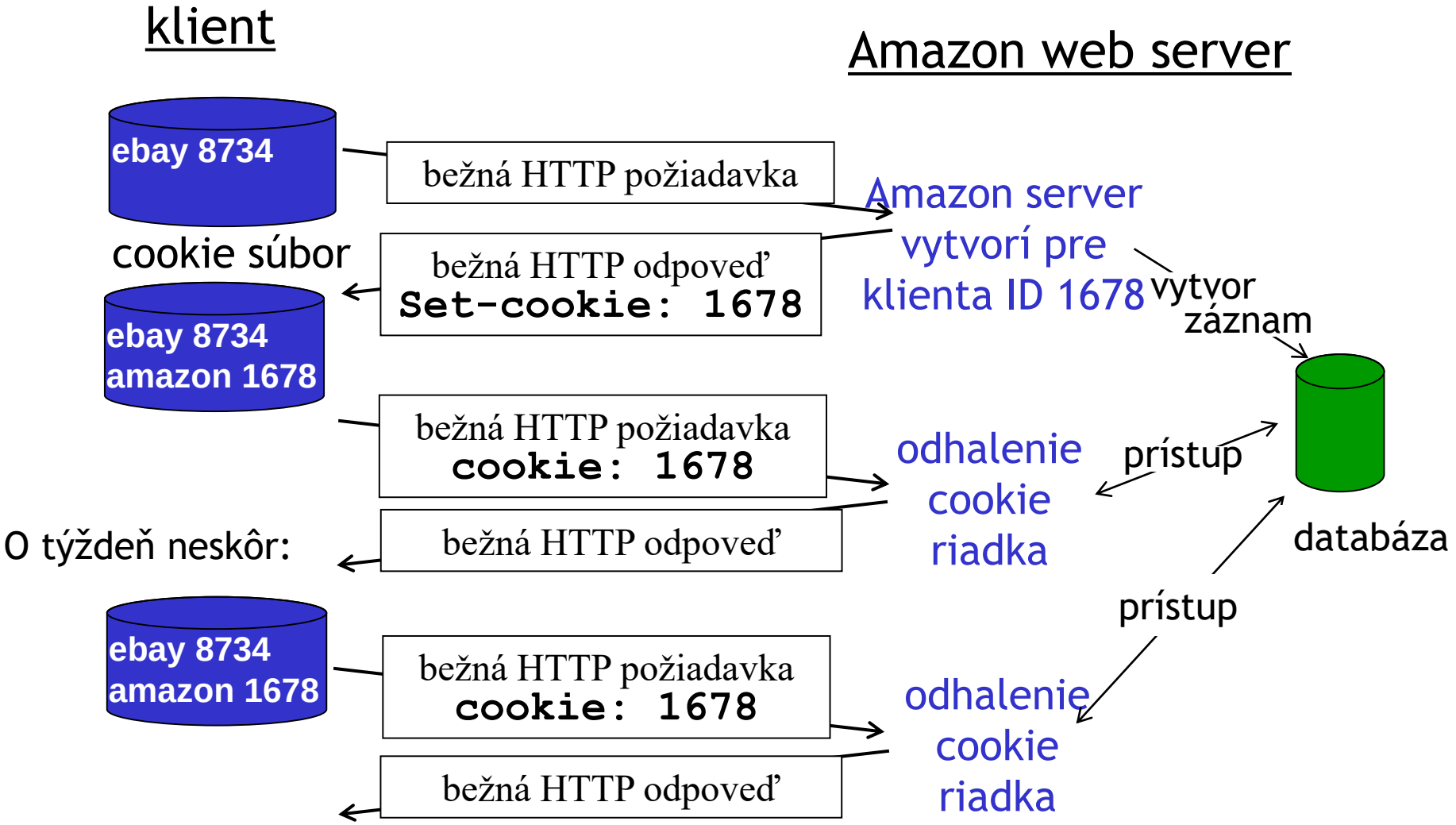
Štyri súčasti:

1) v hlavičke HTTP odpovede servera je cookie riadok
2) aj v hlavičke HTTP požiadavky môže byť cookie riadok
3) browser si uchováva cookie v špeciálnom súbore
4) web server má databázu cookie všetkých komunikácií na nejaký čas dozadu (napr. týždeň aj viac)

Postup:

- Pri prvej HTTP požiadavke server vytvorí
  - jedinečné ID
  - záznam s ID v databáze
- Toto ID sa posiela a uchováva v cookie
- Pri ďalšej návšteve sa môže browser identifikovať s ID v cookie riadku
- Webová aplikácia vie pokračovať v relácii

Stav používateľovej komunikácie: cookies



Cookies

Čo umožňujú:

- autorizáciu
- nákupné košíky
- odporúčania
- stavy komunikácie (napr. webový e-mail klient)

Cookies a súkromie:

- cookies umožňujú web serveru učiť sa správanie používateľa
- Pri nevhodne napísanej web aplikácii sa môžu do cookie správ dostať aj osobné údaje napr. z formulárov

Čo udržiava "stav":

- aplikácie zabezpečujúce fungovanie dynamických webstránok si môžu ku každému cookie ID zvlášť pamätať aktuálny stav aj históriu
- cookies: stav môže byť aj súčasťou cookie v správach (spolu s ID)

DNS: Domain Name System

Ľudské identifikátory:

- Meno, RČ, č. OP, č. pasu

Koncové stanice, routre:

- IP adresa – používaná na adresovanie datagramov na sieťovej vrstve
- "doménové meno", napr. ics.upjs.sk – používané ľuďmi

DNS: preklad z doménových mien na IP adresy a naopak

Čo je DNS?:

- distribuovaná databáza nad celosvetovou hierarchiou mnohých doménových serverov (DNS serverov, name serverov)
- protokol aplikačnej vrstvy umožňujúci koncovým zariadeniam, routrom a DNS serverom komunikovať, aby umožnili preklad doménových mien a IP adries
  - poznámka: adresácia počítačov je záležitosť jadra internetu a predsa je v prípade doménových mien riadená z koncových zariadení

DNS

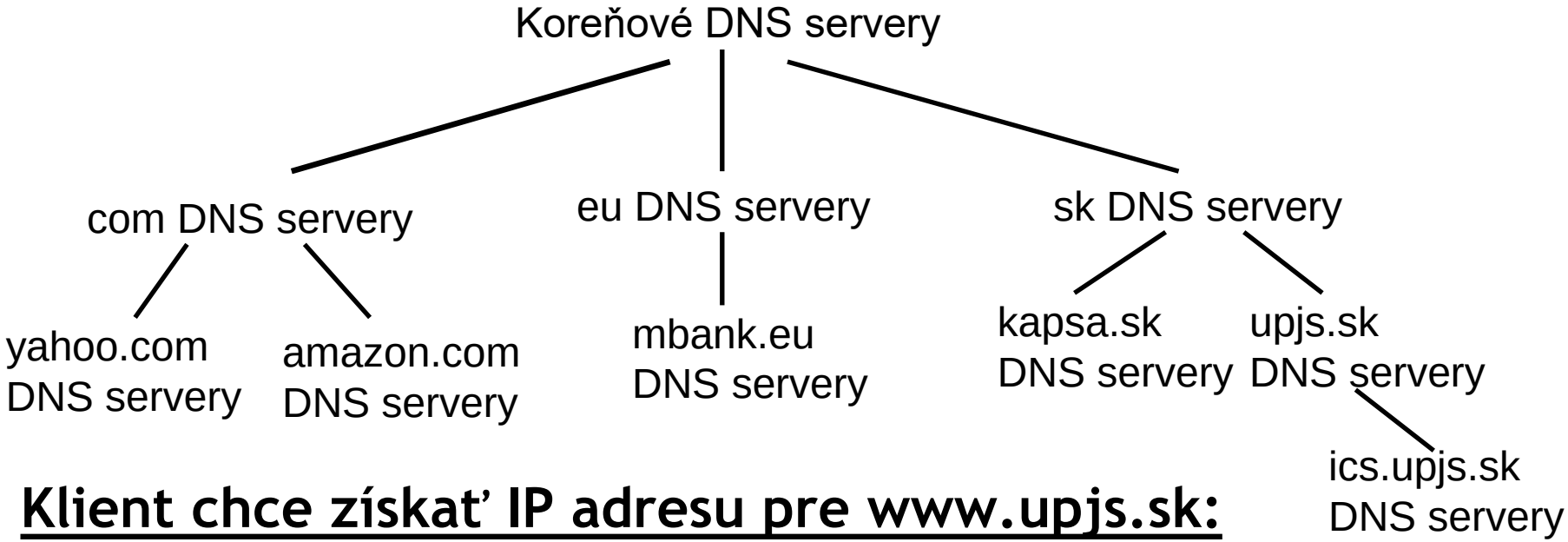
Služby DNS

- preklad doménových mien na IP adresy a späť
- host aliasing
  - Viac mien pre jeden počítač (jednu IP adresu)
- mail server aliasing
  - nezávislé od mena počítača
- distribuovanie záťaže
  - replikované webové servery: viac IP adries pre jedno doménové meno

Prečo nie je DNS centralizované?

- single point of failure
  - ak by centrálne DNS zlyhalo, "skolabuje" celý Internet
- veľká záťaž siete na jednom mieste
- vzdialenosť centrálnej databázy
- spravovanie

Distribuovaná, hierarchická databáza



Klient chce získať IP adresu pre www.upjs.sk:

- klient sa opýta koreňového DNS servera, ten mu odpovie, kde sídli DNS server pre doménu sk
- klient sa opýta DNS servera pre doménu sk, ten mu odpovie, kde sídli DNS server pre doménu upjs.sk
- klient sa opýta DNS servera pre doménu upjs.sk, ten mu povie mu IP adresu pre www.upjs.sk

DNS: Koreňové doménové servery

- Kontaktované lokálnym doménovým serverom, ak nevie sám odpovedať
- Na svete je presne 13 IP adries, na ktorých sú koreňové DNS servery
  - Hovoríme, že je 13 koreňových doménových serverov
  - Reálne takmer každý z nich sídli na mnohých miestach
    - Veľa počítačov s rovnakou IP adresou (viď. anycast)
    - http://www.root-servers.org/

TLD a autoritatívne servery

- Top-level domain (TLD) servery:
  - Zodpovedné za com, org, net, edu, atď. A za všetky koncovky krajín sk, cz, at, uk, fr, ca, jp, ...
  - Doménu **com** spravuje Network Solutions
  - Doménu **sk** spravuje štátna spoločnosť SK-NIC
- Autoritatívne DNS servery:
  - DNS servery organizácií (napr. doménový server pre upjs.sk)
  - spravujú mená počítačov pre danú doménu
  - ich IP sú evidované v nadradených DNS serveroch (napr. TLD)
  - opak, teda neautoritatívne servery neexistujú
- existujú neautoritatívne odpovede od DNS serverov, ktoré si pamätajú preklad daného mena a IP adresy z predchádzajúcej komunikácie vo svojej cache

Lokálne doménové servery

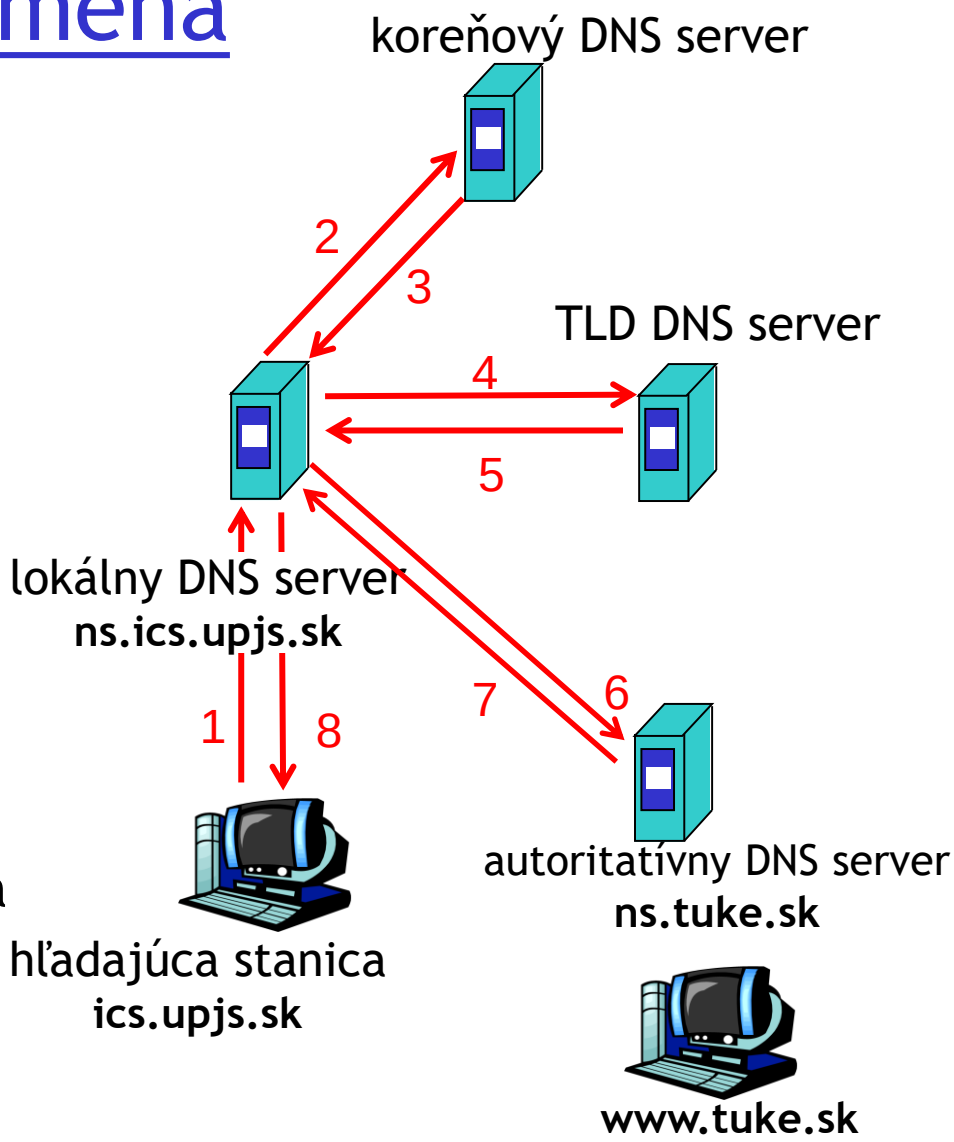
- nemusia byť zakomponované v hierarchii
- každý provider nejaký lokálny doménový server prevádzkuje
  - označuje sa ako “default name server”
- keď klient zadá požiadavku pre DNS, tento server komunikuje s ostatnými doménovými servermi a nakoniec vráti výsledok
  - niečo podobné ako proxy web server

Príklad prekladu mena na IP adresu

- Proces na ics.upjs.sk chce IP adresu pre www.tuke.sk

Postupné otázky:

- oslovený server odpovie menom servera, ktorý treba osloviť ako ďalší
- "Neviem odpovedať, spýtaj sa tohoto servera"
- "Ja ho nemám, ja ho nemám, má ho číslo 195.12.159.3"



DNS: cache-ovanie a obnovovanie záznamov

- keď sa nejaký server dozvie mapovanie z iného servera, tak si ho zapamätá v *cache*
  - záznamom v cache po čase vyprší platnosť
  - typicky blízke TLD servery sú známe lokálnym doménovým serverom
    - Teda sa nemusia zakaždým pýtať koreňových DNS serverov, kde sú TLD DNS servery

DNS záznamy: RR (resource records)

DNS: distribuované úložisko DNS záznamov

Formát RR: **(meno, hodnota, typ, ttl)**

Time To Live
=
dĺžka života

- Typ A
  - **meno** je doménové meno
  - **hodnota** je IPv4 adresa
- Typ AAAA
  - **to isté pre IPv6 adresy**
- Typ NS
  - **meno** je doména (napr. upjs.sk)
  - **hodnota** je meno autoritatívneho servera pre túto doménu
- Typ CNAME
  - **meno** je alias pre "kánonické" (skutočné) meno

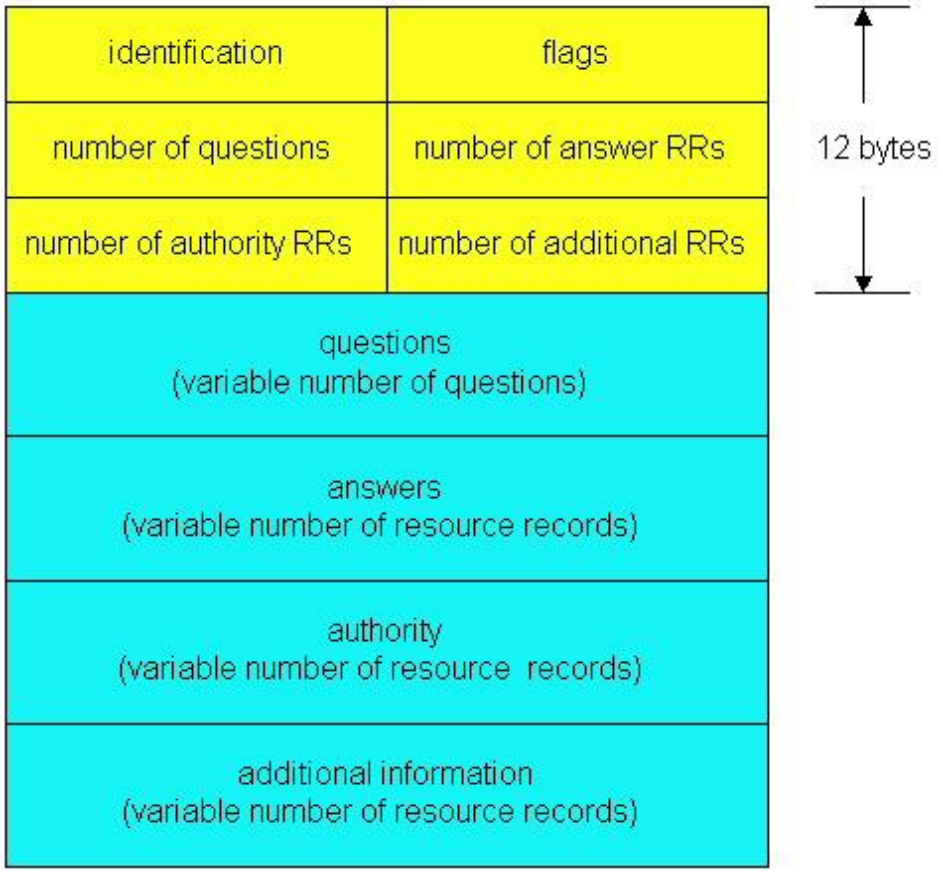
    www.ibm.com je v skutočnosti servereast.backup2.ibm.com
  - **hodnota** je kánonické meno
- Typ MX
  - **hodnota** je meno mailového servera asociovaného s **menom**

DNS protokol: RFC 1035

Požiadavky aj *odpovede* majú rovnaký formát – binárny

Hlavička správy:

- identification: náhodné 16 bitové číslo spoločné pre otázku a odpoveď
- flags (príznaky):
  - požiadavka alebo odpoveď
  - žiadosť o rekurziu
  - možnosť rekurzie
  - autoritatívna odpoveď alebo nie



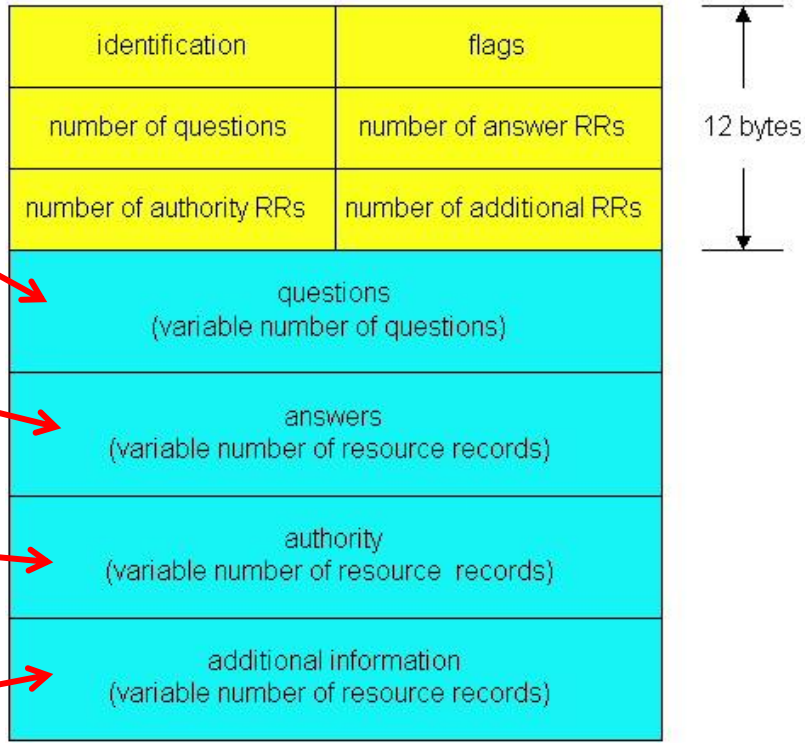
DNS protokol

meno a typ v otázke

RR záznamy v odpovedi

záznamy autoritatívnych serverov

dodatočné záznamy



DNS server

- počúva na porte 53
- komunikácia cez TCP alebo UDP
  - tí, čo žiadajú o preklad, obvykle komunikujú cez UDP
  - zrkadlá žiadajúce o všetky autoritatívne informácie komunikujú cez TCP
- ak chcem nový autoritatívny DNS server pre nejakú doménu, napríklad pre **psin.sk**, musí prevádzkovateľ TLD DNS servera pre doménu sk vložiť nasledovné údaje:

```
(psin.sk, ns.psin.sk, NS)
(ns.psin.sk, 10.10.10.10, A)
```

- Na serveri ns.psin.sk si môžem vytvárať záznamy typu A, napríklad www.psin.sk, alebo aj MX záznamy, ak chcem prevádzkovať aj mailový server

Zdieľanie súborov cez P2P

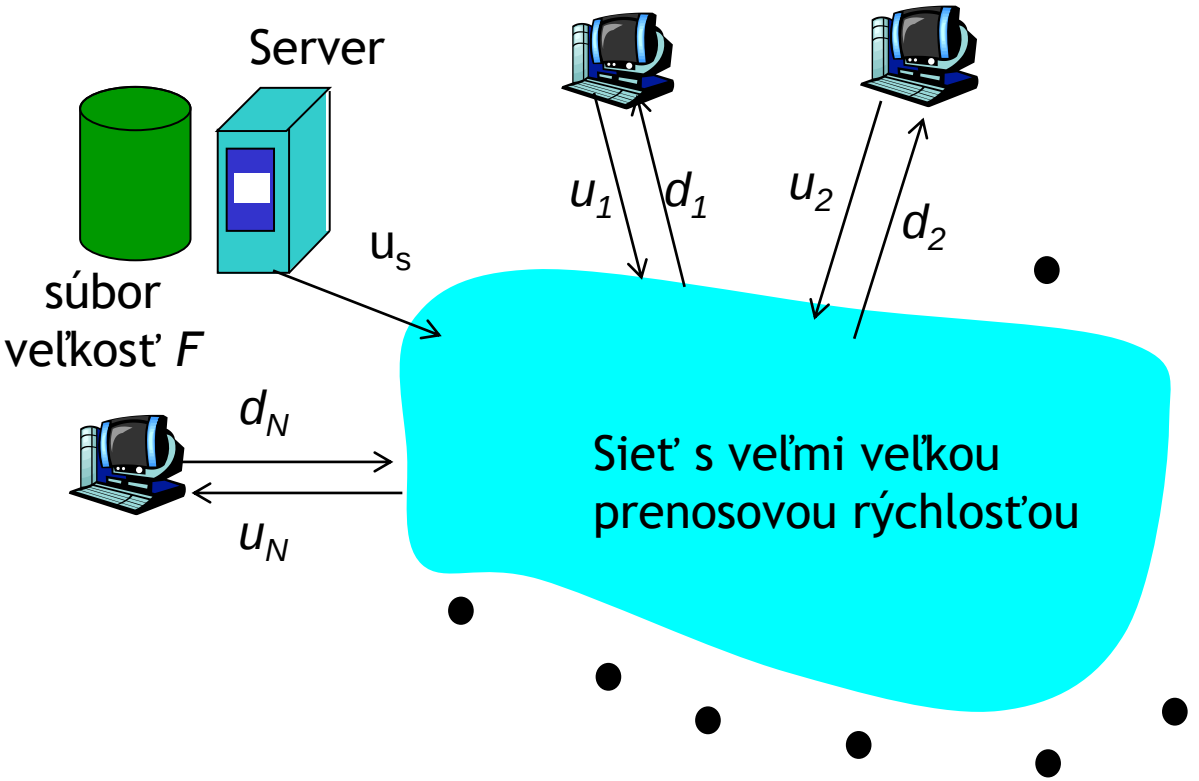
Príklad

- Alica spúšťa P2P program na svojom notebooku
- bežne sa pripája na rôznych miestach a stále dostane inú IP adresu
- hľadá film “Matrix”
- aplikácia jej zobrazí, ktorý z peerov v jej P2P sieti zdieľa film Matrix
- Alica si vyberie spomedzi peerov Boba
- súbor sa kopíruje z Bobovho počítača do Alicinho notebooku
- zatiaľ čo Alica sťahuje film Matrix, iní peerovia môžu ťahať zdieľané dáta z jej notebooku
- Alicin P2P program je zároveň klientom aj serverom.

Všetci peerovia sú servery = vysoká škálovanosť!

Porovnanie klient/server a P2P architektúry

Otázka : Koľko času je potrebné na distribúciu súboru pôvodne z jedného servera na *N* počítačov?



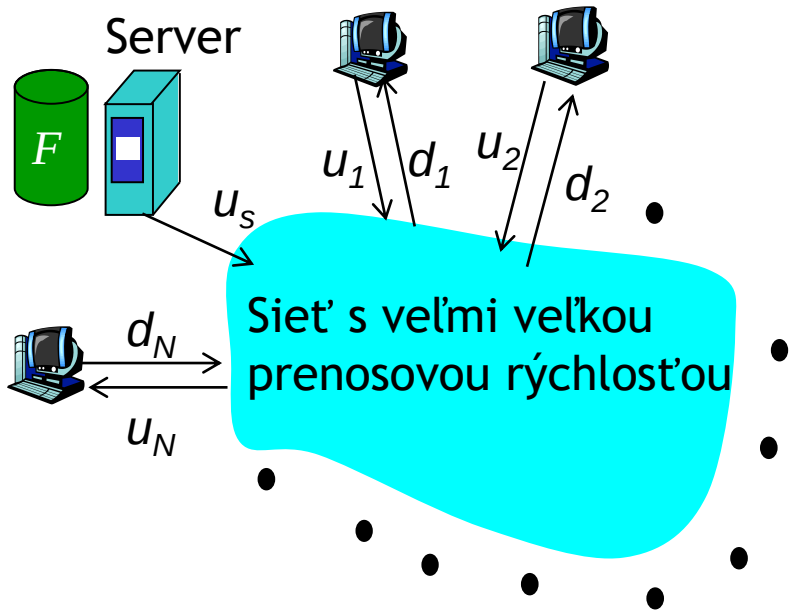
u_s: šírka pásma servera pre upload

u_i: šírka pásma klienta alebo peera pre upload

d_i: šírka pásma klienta alebo peera pre download

Klient/server: čas distribúcie súboru

- server odosielajúci *N* kópií súboru veľkosti *F* sekvenčne:
  - NF/u_s sekúnd
- Klient *i* potrebuje minimálne F/d_i sekúnd na download

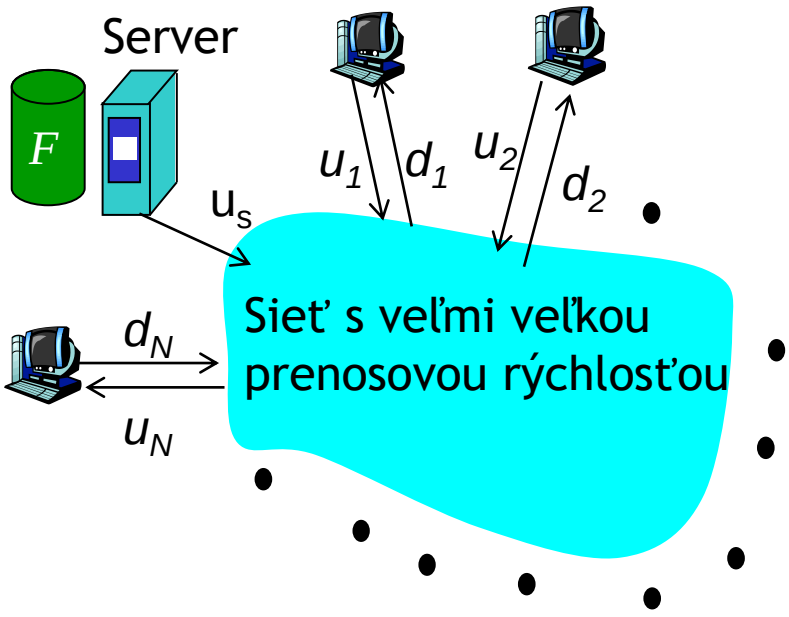


Čas na rozposlanie súboru veľkosti *F* ku *N* klientom = $d_{ks} = \max \{ NF/u_s, F/\min_i(d_i) \}$

pre veľké N rastie lineárne

P2P: čas distribúcie súboru

- server musí poslať minimálne jednu kópiu: F/u_s sekúnd
- Klient *i* potrebuje minimálne F/d_i sekúnd na download
- NF bitov musí byť stiahnutých na peerov dokopy



- Najrýchlejšie odosielanie zo všetkých peerov súčasne tvorí šírku pásma $u_s + \Sigma u_i$

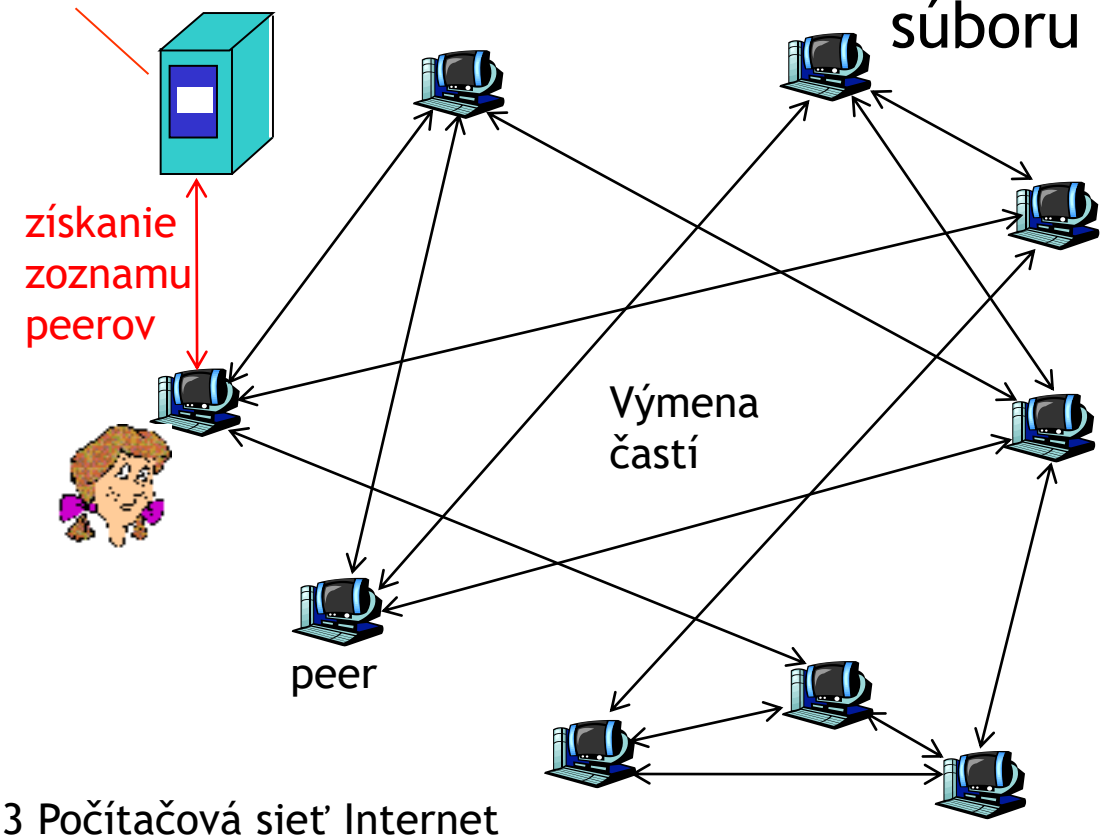
$$d_{P2P} = \max \{ F/u_s, F/\min_i(d_i) , NF/(u_s + \sum_i u_i) \}$$

BitTorrent

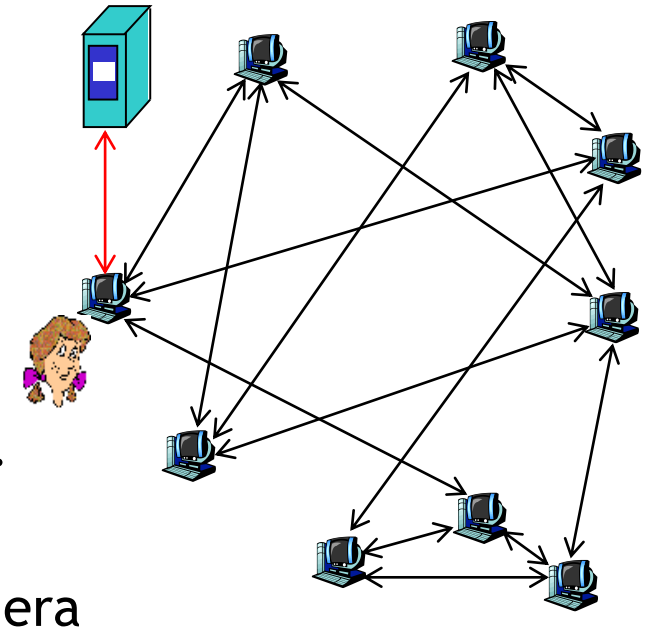
- distribúcia súborov cez P2P

tracker: registruje peerov zúčastnených na distribúcii

torrent: skupina peerov vymieňajúca si časti súboru



BitTorrent



- súbor je rozdelený na malé časti
- = *chunks* *(mocnina 2, typicky 256 KB – 4 MB).*
- peer sa napojí na torrent:
  - nemá žiadne časti, ale časom si nejaké zozbiera
  - je registrovaný u trackera na získanie zoznamu peerov, na ktorých sa napojí
  - alternatívne cez peer exchange/local peer discovery
- počas sťahovania, peer posiela na požiadanie tie časti, ktoré už má, iným peerom
- peerovia sa môžu pripájať a odpájať
- keď peer doťahal celý súbor, môže sa (sebecky) odpojiť alebo (obetavo) ostať napojený = seeder

BitTorrent

Výber častí na sťahovanie:

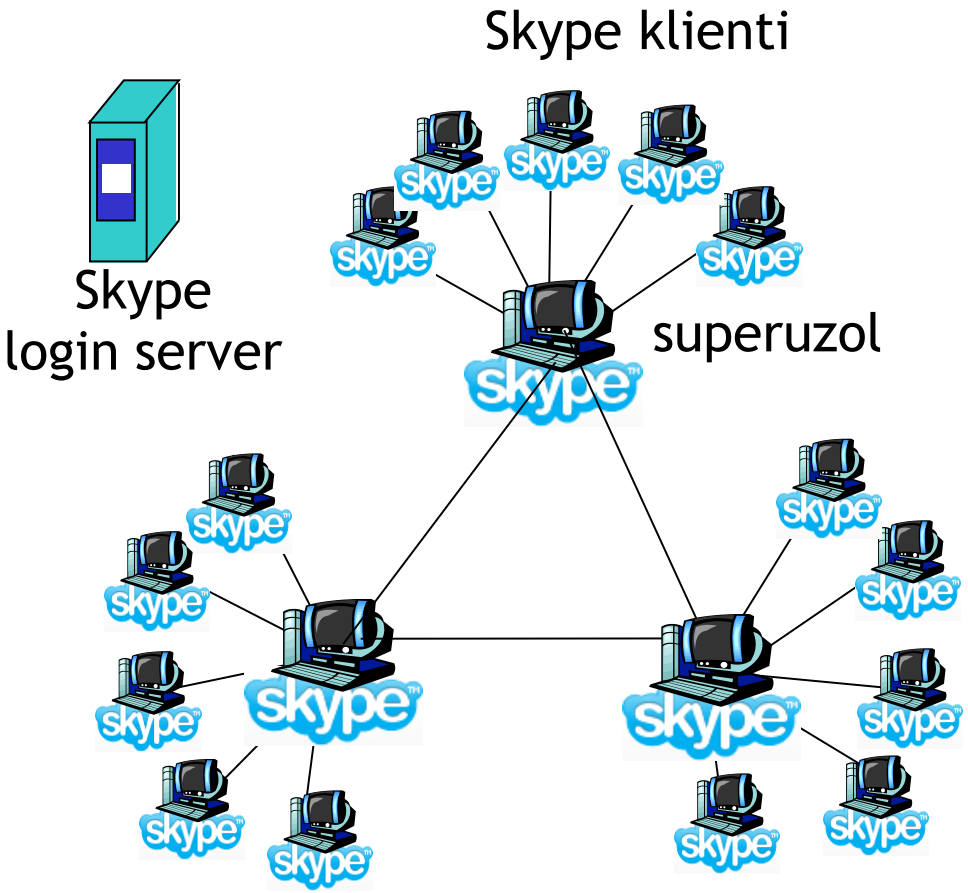
- V ľubovoľnom čase majú peerovia rôzne podmnožiny častí
- Peer (Alica) stále dookola žiada svojich susedov o zoznam častí, ktoré majú
- Alica generuje žiadosti o tie časti, ktoré zatiaľ nemá
  - najskôr najzriedkavejšie

Odosielanie častí (príklad):

- Alica odosiela časti štyrom žiadajúcim susedom, ktorí jej posielajú svoje časti *najvyššou rýchlosťou*
  - Prepočítava najlepších 4 každých 10 sekúnd
- každých 30 sekúnd si vyberie jedného náhodného žiadateľa a pošle mu, čo žiadal
  - Tento peer sa môže dostať medzi najlepších štyroch

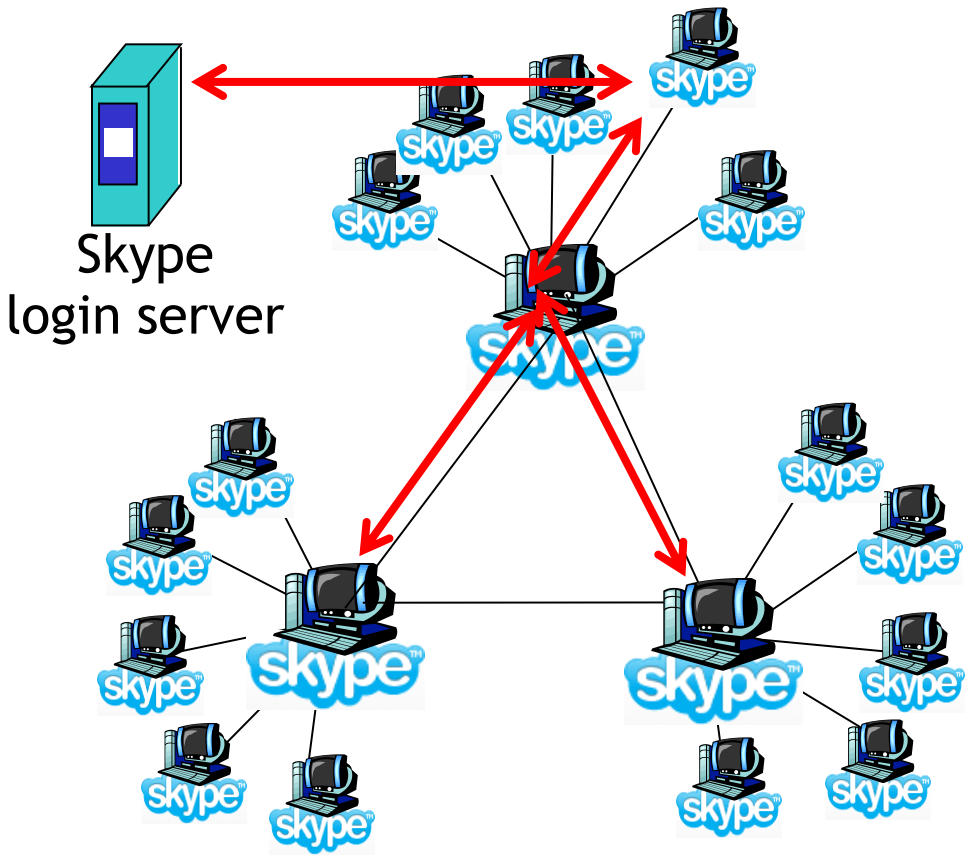
Skype

- P2P (pc-to-pc, pc-to-phone, phone-to-pc) Voice-Over-IP (VoIP) aplikácia
  - tiež Instant Messaging
- neverejný protokol aplikačnej vrstvy (všetko čo vieme, je odpozorované)
- reverzným inžinieringom boli získané zdrojové kódy
- hierarchická sieť



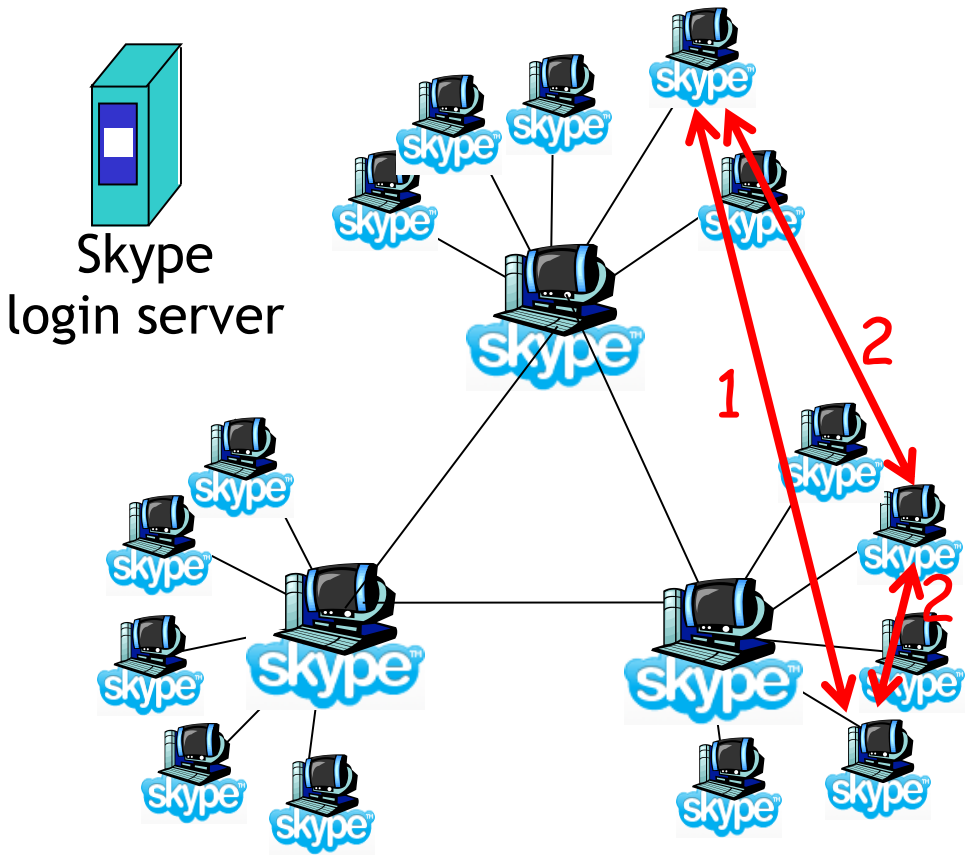
Skype

- Peer sa autentifikuje voči login serveru,
- zaregistruje sa u blízkeho superuzla,
- jeho superuzol ho informuje o jeho on-line kontaktoch,
- na superuzloch je (pravdepodobne) distribuovaný index kontaktov.



Skype

- telefonát: cieľový peer je informovaný jeho superuzlom
- ak jeden z peerov má verejnú IP adresu, napoja sa **priamo** a komunikujú medzi sebou (1)
- ak nie, superuzol vyberie **sprostredkovateľa** spomedzi peerov s verejnou IP, obaja telefonujúci sa naňho napoja cez TCP kanál a celá komunikácia ide cez neho (2)

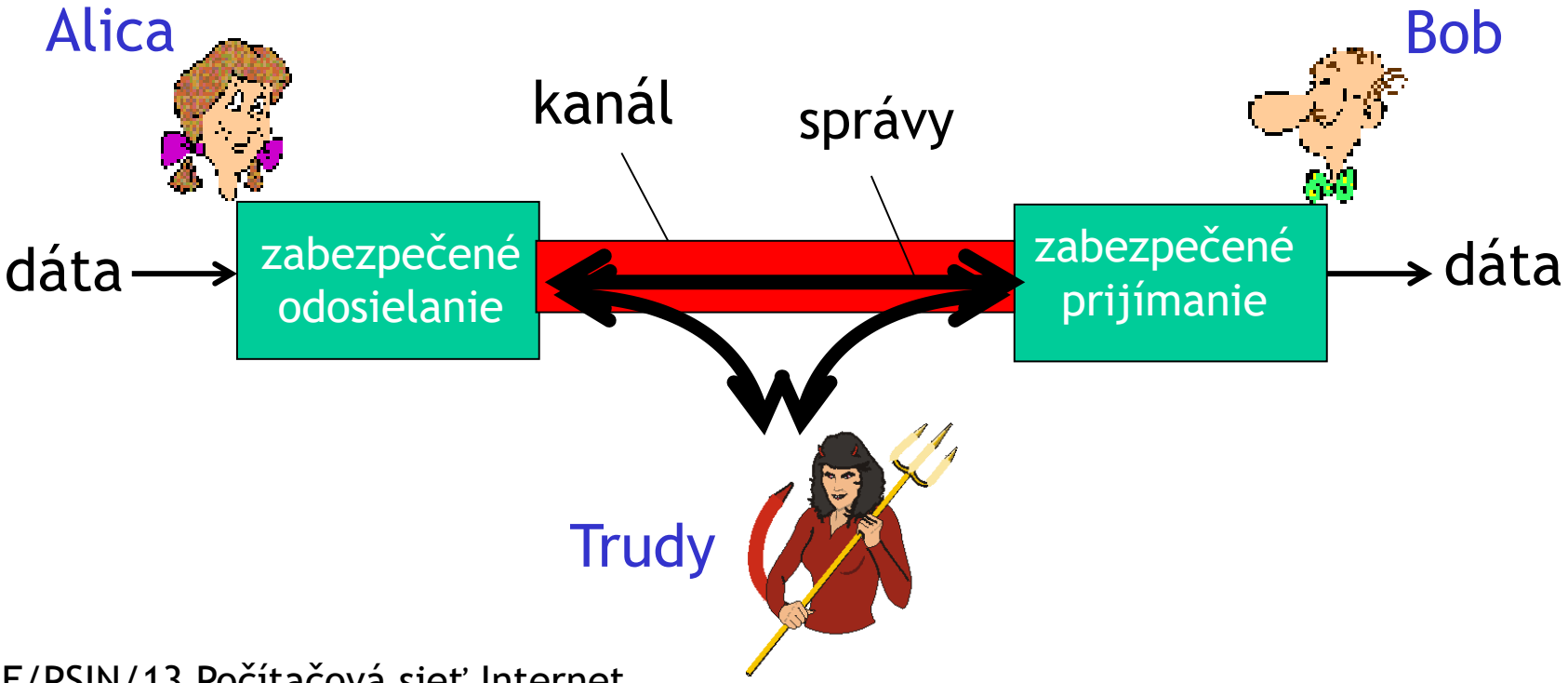


Zabezpečenie protokolov

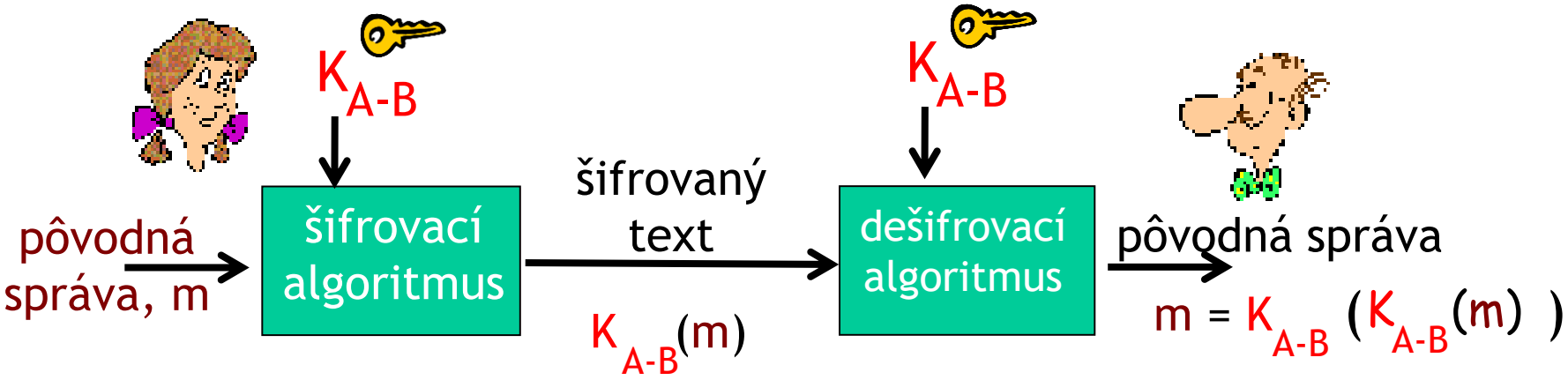
- zadefinujeme dôležité pojmy
  - symetrické šifrovanie
  - asymetrické šifrovanie, súkromný a verejný kľúč
  - elektronický podpis
  - certifikačná autorita
- príklad zabezpečenia pre aplikačné protokoly
- ...iba “slabý vývar” bezpečnosti na internete

Priatelia a útočníci: Alica, Bob, Trudy

- známe mená vo svete bezpečnosti sietí
- Alica a Bob chcú "bezpečne" komunikovať
- Trudy (útočník) môže odpočúvať, meniť, mazať a pridávať správy do ich spojenia

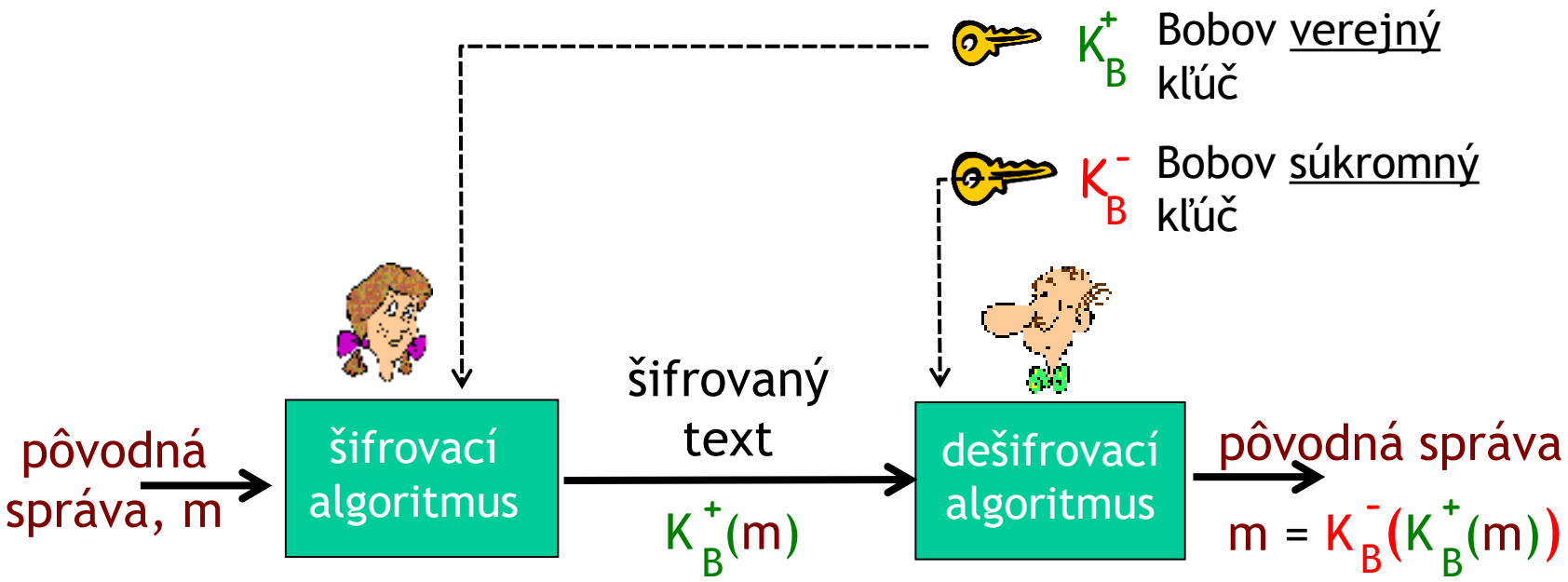


Symetrická kryptografia



- Symetrická kryptografia: Alica a Bob zdieľajú rovnaký symetrický kľúč: K_{A-B}
- Otázka: Ako si Alica a Bob dohodnú spoločný kľúč, ak ho zatiaľ dohodnutý nemajú?

Asymetrická kryptografia



Vlastnosti súkromného z verejného kľúča

Požiadavky:

1. Potrebujeme $K_B^+(\ )$ a $K_B^-(\ )$ také, že

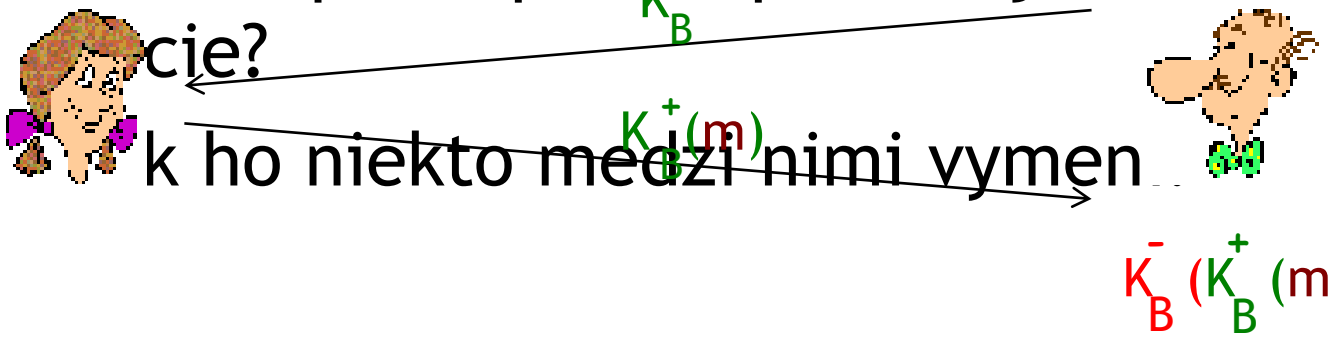
$$K_B^-(K_B^+(m)) = m$$

2. Ak máme verejný kľúč K_B^+, malo by byť nemožné vypočítať z neho súkromný kľúč K_B^-

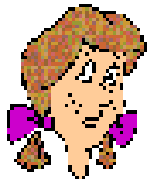
Napr. pre algoritmus RSA: Rivest, Shamir, Adleman platí aj užitočný opačný vzťah: $K_B^+(K_B^-(m)) = m$

Problém distribúcie verejného kľúča

- Ako si má Alica bezpečne stiahnuť Bobov verejný kľúč?
  - Keďže je verejný, Bob ho môže dať hocikomu - ak ho niekto iný prečíta, tak to nevadí
  - Zavesí ho na web?
  - Pošle ho ako prvú správu spoločnej komu cie?
  - Ale k ho niekto medzi nimi vymen



Man-in-the-middle



K_B^+

$K_B^+(m)$

$K_B^-(K_B^+(m)) = m$

K_B^+

K_T^+

$K_T^+(m)$

$K_T^-(K_T^+(m)) = m$

$K_B^+(m)$

$K_B^-(K_B^+(m)) = m$

Šifrovacia hash funkcia

- vezmeme ľubovoľne dlhú správu m, vygenerujeme hodnotu H(m) fixnej dĺžky, ktorú následne môžem zašifrovať s použitím kľúča K
- vlastnosti šifrovacej hash funkcie:
  - malo by byť výpočtovo ťažké nájsť dve rôzne správy x a y také, že H(x) = H(y)
  - na základe znalosti H(m) by sme nemali vedieť získať m

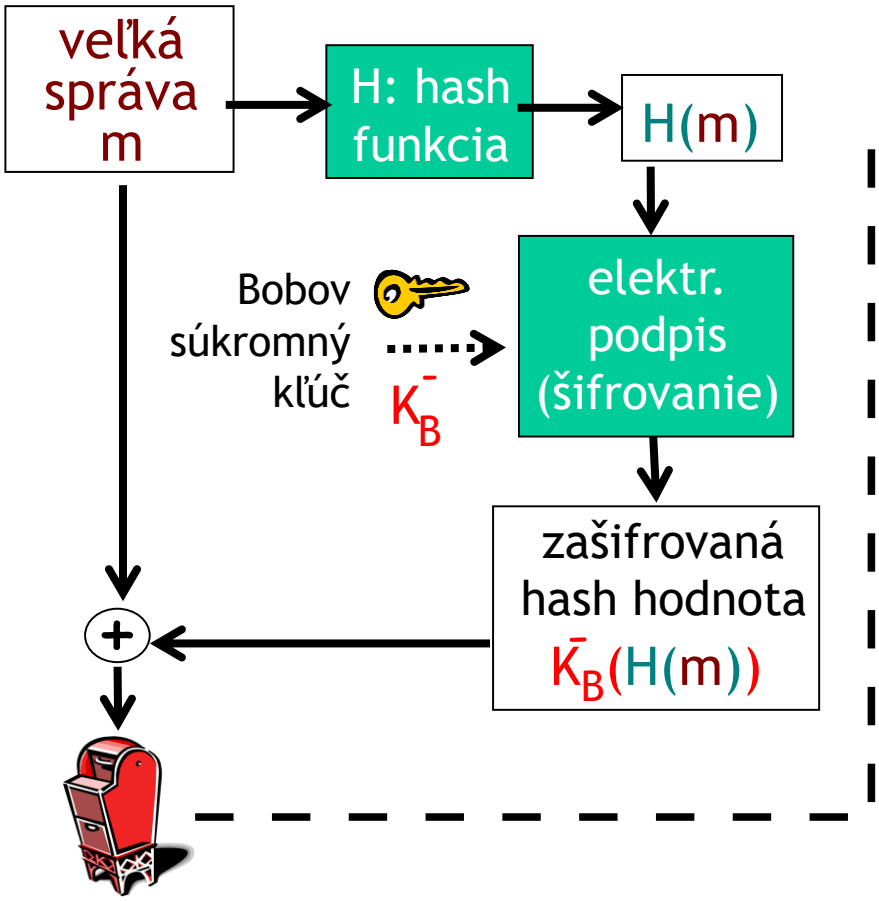
Elektronický podpis

šifrovacia technika analogická s ručným podpisom

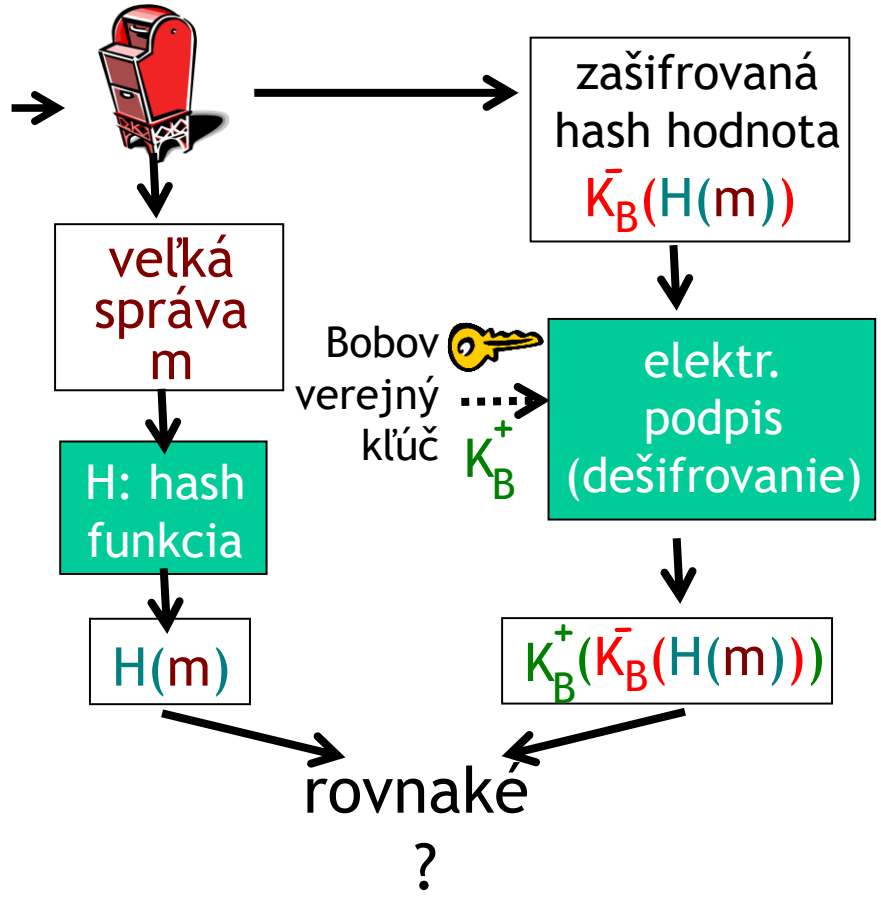
- odosielateľ (Bob) elektronicky podpíše jeho dokument deklarujúc, napr. že je autorom dokumentu
- podpis sa prikladá k pôvodnému (nezašifrovanému) dokumentu
- verifikovateľnosť: vieme dokázať, že dokument je podpísaný Bobovým podpisom
- nepopierateľnosť: vieme tvrdiť, že podpis tohto dokumentu nemohol byť vygenerovaný niekým iným ako Bobom

Elektronický podpis

Bob odošle elektronicky podpísanú správu:



Alica overí pravosť podpisu a neporušenie elektronicky podpísanej správy:



Certifikovanie verejného kľúča

problém s verejným kľúčom:

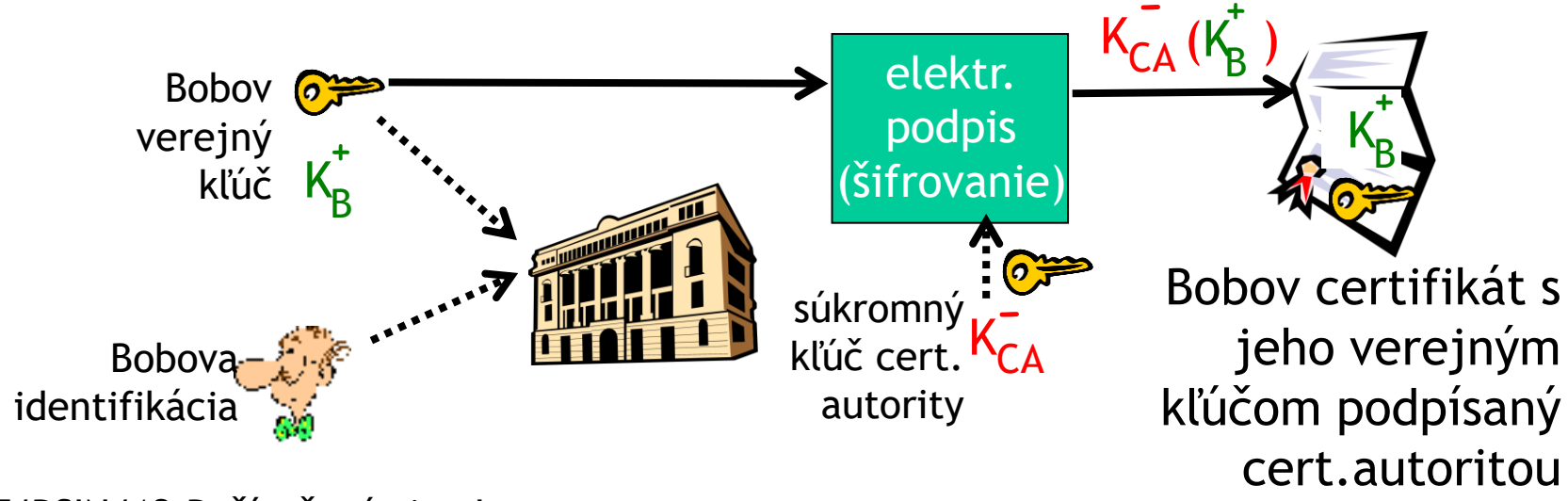
- Keď Alica získa Bobov verejný kľúč (z webstránky, e-mailu), ako môže *vedieť*, že je to Bobov a nie Trudyn verejný kľúč?

riešenie:

- dôveryhodná certifikačná autorita (CA)

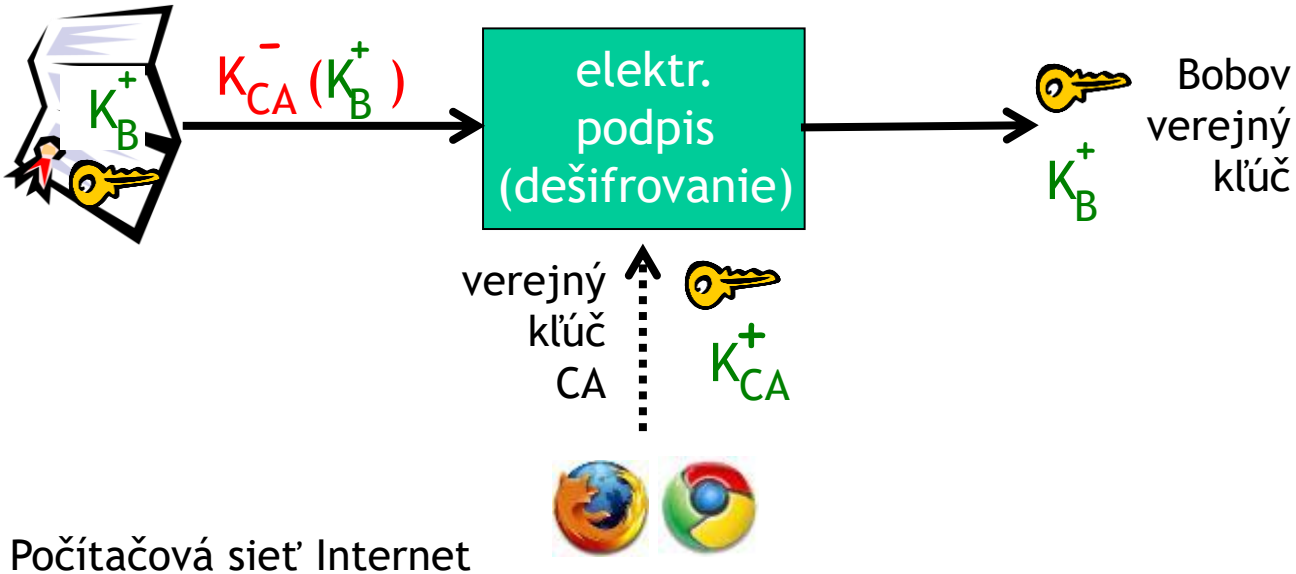
Certifikačné autority

- Certifikačná autorita (CA): spája entitu E (počítač, osobu) s jej verejným kľúčom.
- E si registruje svoj verejný kľúč u CA.
  - E poskytne "dôkaz identity" pre CA.
  - CA vytvorí certifikát spájajúci E a jeho verejný kľúč
  - certifikát je elektronicky podpísaný certifikačnou autoritou, ktorá tvrdí: "Toto je verejný kľúč patriaci E."



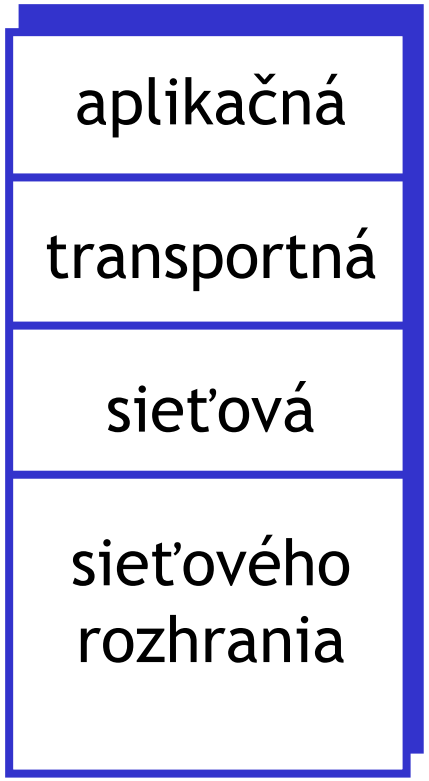
Certifikačné autority

- Keď Alica chce získať Bobov verejný kľúč:
  - vezme si Bobov certifikát (od Boba alebo inak).
  - použije verejný kľúč certifikačnej autority a získa tak overený Bobov verejný kľúč
  - verejné kľúče dôveryhodných CA bývajú súčasťou inštalácie sieťových programov (napr. web. prehliadačov)



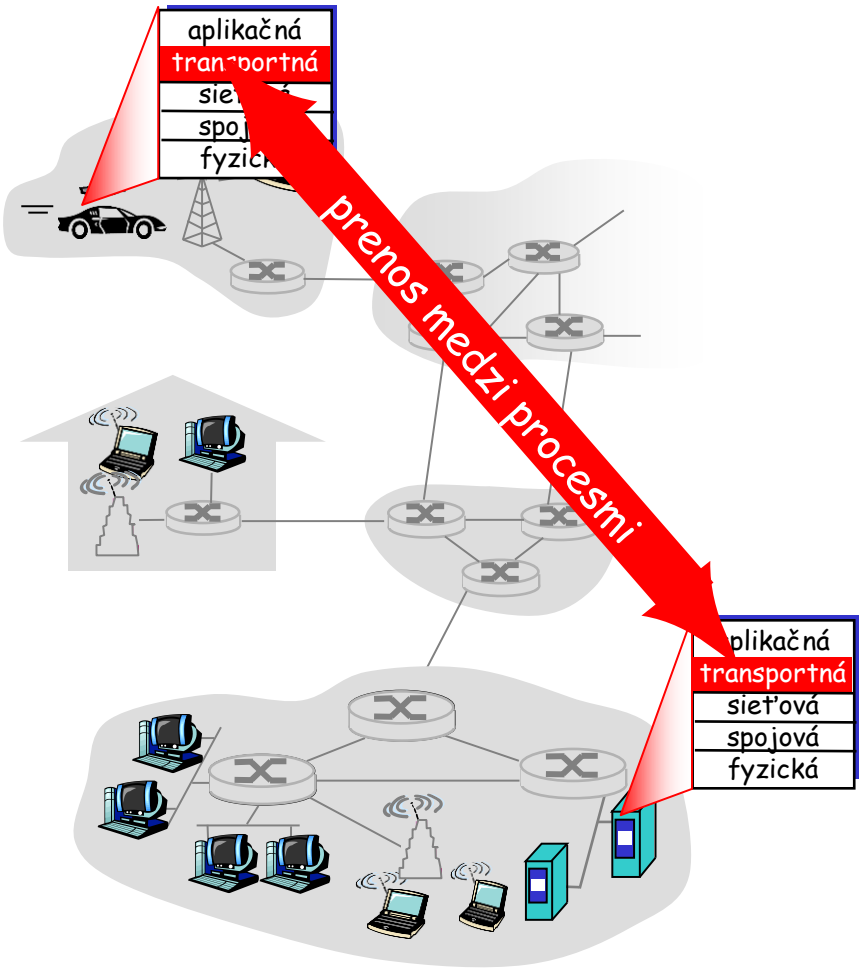
Implementácia internetu TCP/IP

- aplikačná (application): umožňuje fungovanie sieťových aplikácií – definuje tvar a poradie správ
  - prezentačná a relačná splynuli s aplikačnou
  - tieto služby musí aj tak mať implementované aplikácia, ak to potrebuje
  - a čo ak nepotrebuje?
  - HTTP, FTP, SMTP, POP, IMAP, XMPP, SSH, ...
- transportná (transport): prenáša dáta medzi dvoma procesmi na rôznych koncových zariadeniach
  - TCP, UDP
- sieťová (network): smeruje datagramy od odosielateľa k príjemcovi
  - IP, smerovacie protokoly
- sieťového rozhrania (network interface): splynutie funkcionality do technológií na prenos dát medzi susednými sieťovými prvkami a spôsobu prenášania binárnych dát
  - PPP, Ethernet



Protokoly a služby transportnej vrstvy

- poskytujú logický prenos dát medzi procesmi programov bežiacich na vzdialených koncových zariadeniach
- transportné protokoly sa realizujú iba na koncových staniciach
  - **odosielajúca strana:** rozdeľuje správy odosielané z procesu a vkladá ich do segmentov a tie posiela na spracovanie sieťovej vrstve
  - **prijímajúca strana:** spája dáta zo segmentov prijaté zo sieťovej vrstvy do správ a posiela ich procesu v aplikačnej vrstve
- je niekoľko transportných protokolov:
  - štandardne na internete: TCP a UDP



Transportná verzus sieťová vrstva

- *transportná vrstva:* umožňuje logickú komunikáciu medzi procesmi
  - spolieha sa na služby sieťovej vrstvy
- *sieťová vrstva:* umožňuje logickú komunikáciu medzi stanicami
  - transportná vrstva iba povie, ktorej cieľovej stanici je segment určený

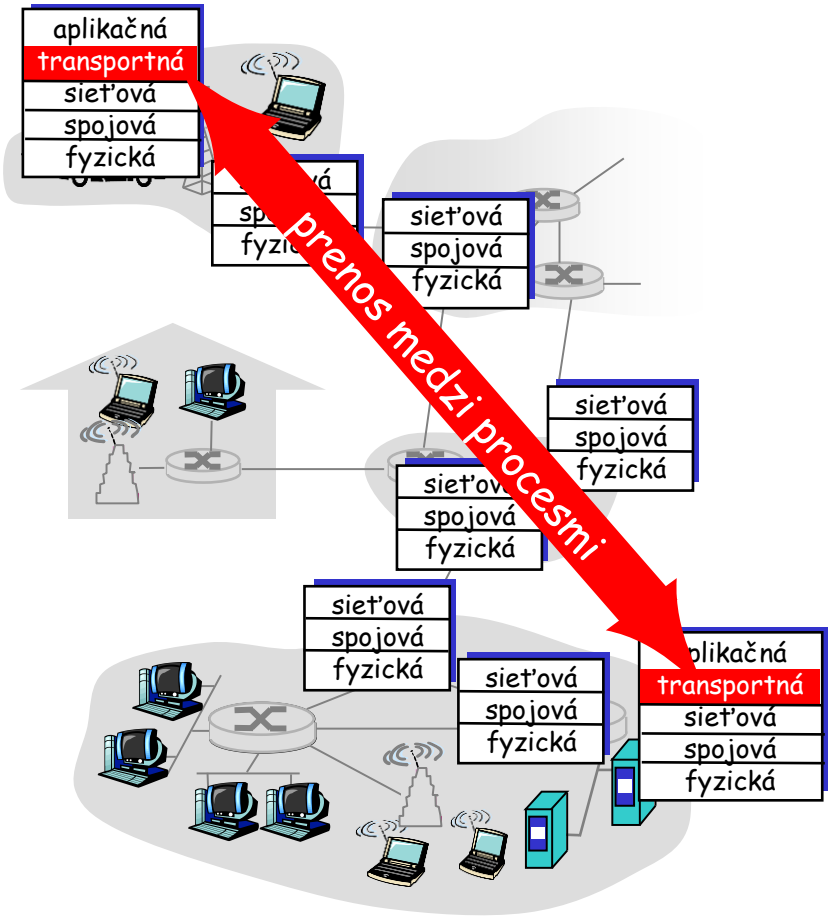
Analógia "Koncert Madonny"

Treba dopraviť pódium a techniku z Londýna do Košíc

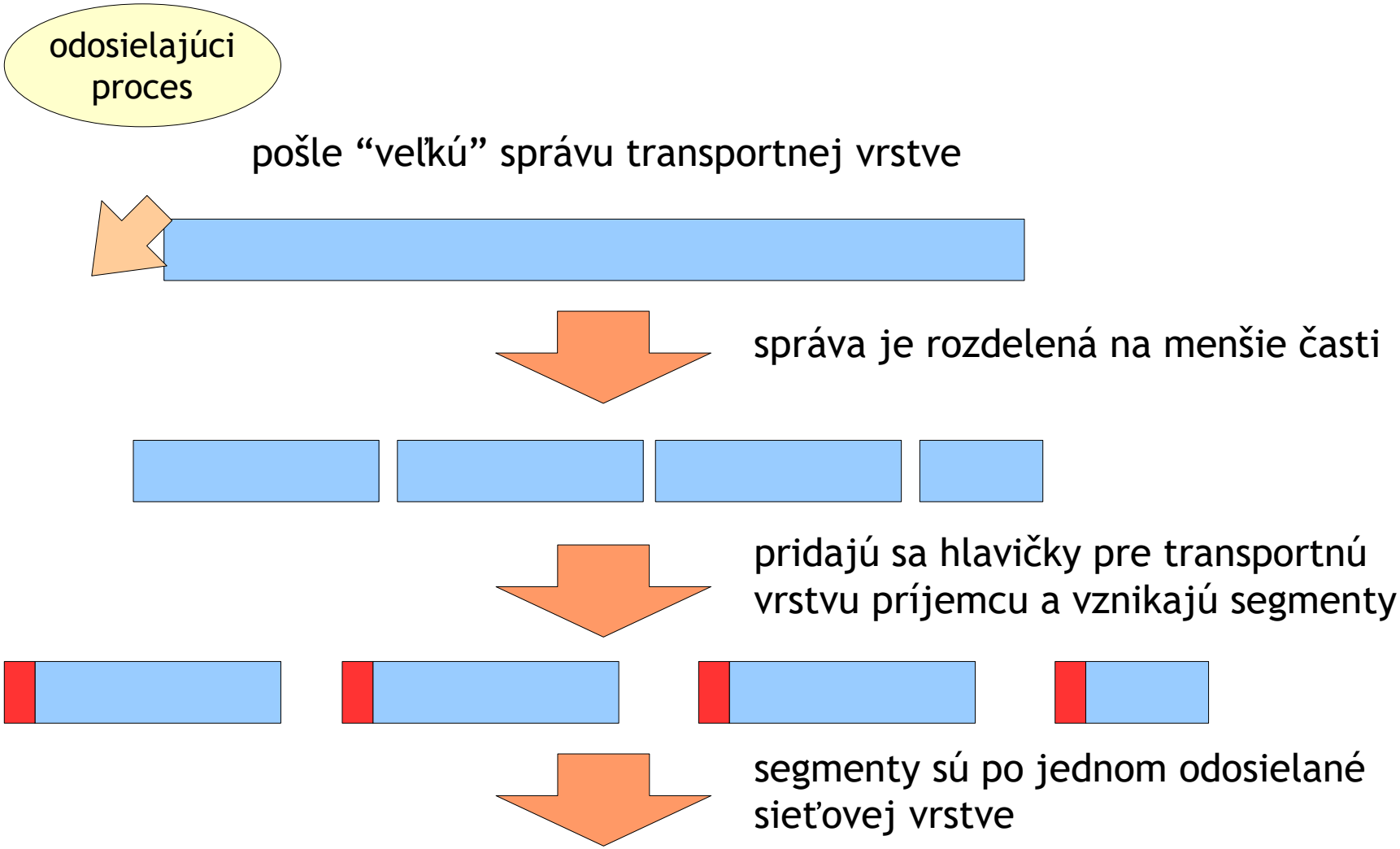
- procesy = londýnsky a košický manažér
- správy aplikačnej vrstvy = pódium a technika
- koncové stanice = mestá
- transportný protokol = tímy technikov v koncových mestách a komunikácia medzi nimi
- protokol sieťovej vrstvy= zásielková služba (64 kamiónov, 2 lode a logistika)

Internetové protokoly transportnej vrstvy

- potvrdzované doručenie v správnom poradí: TCP
  - potvrdzovanie segmentov
  - kontrola zahltenia
  - kontrola toku dát
  - nastavovanie spojenia
- nepotvrdzované doručenie v "náhodnom" poradí: UDP
  - žiadne extra vylepšenie oproti spoľahlivosti IP
- neposkytované služby:
  - zabezpečenie času doručenia
  - zabezpečenie prenosovej rýchlosti



Delenie správ do segmentov



Delenie a spájanie správ

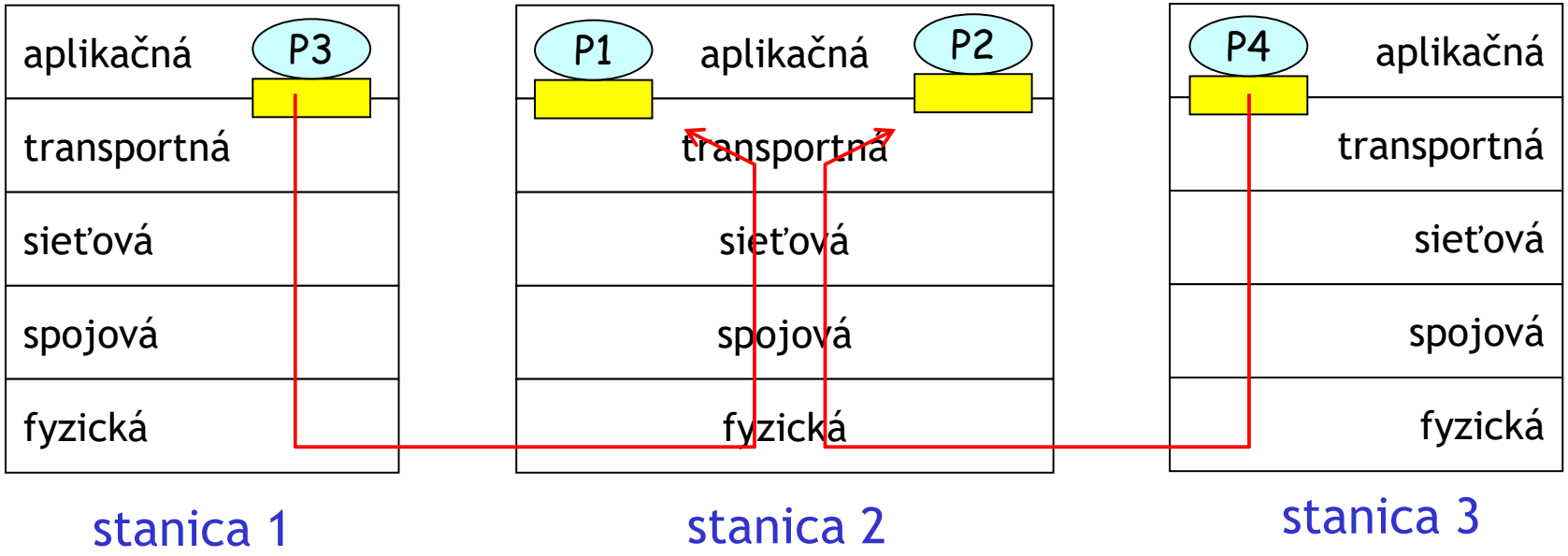
U odosielateľa:

zozbieranie správ zo soketov, vytvorenie segmentov a odoslanie príjemcovi

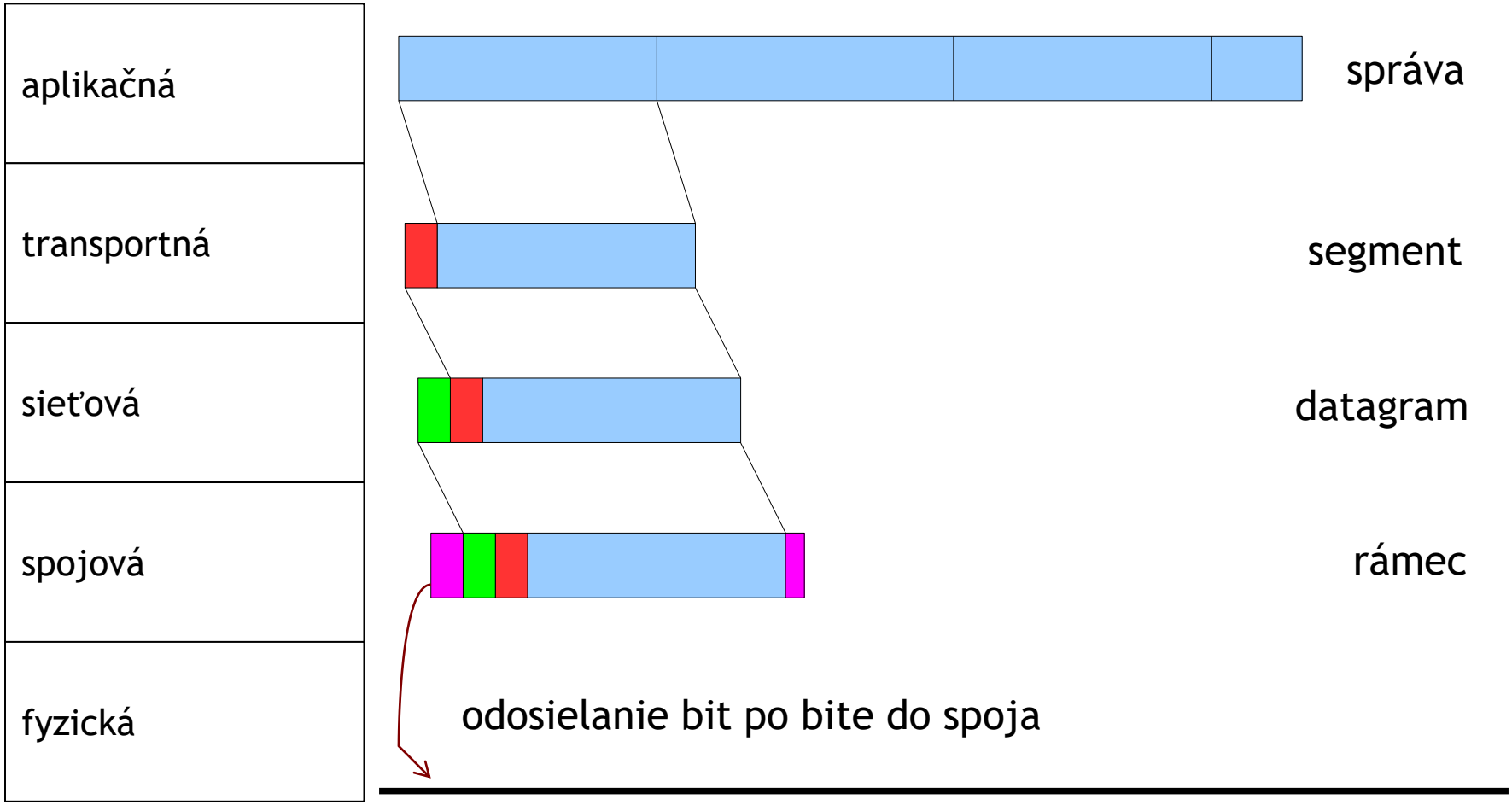
U príjemcu:

vyskladanie správ z prijatých segmentov a ich doručenie k správnym soketom

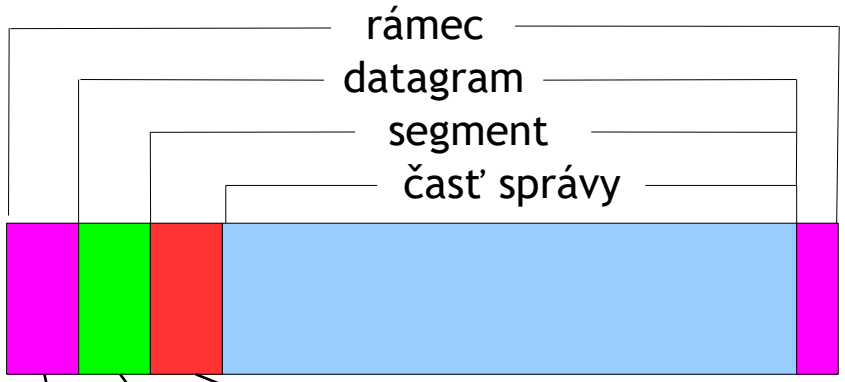
= soket = proces



Tvorba paketu



Adresovacie údaje v pakete



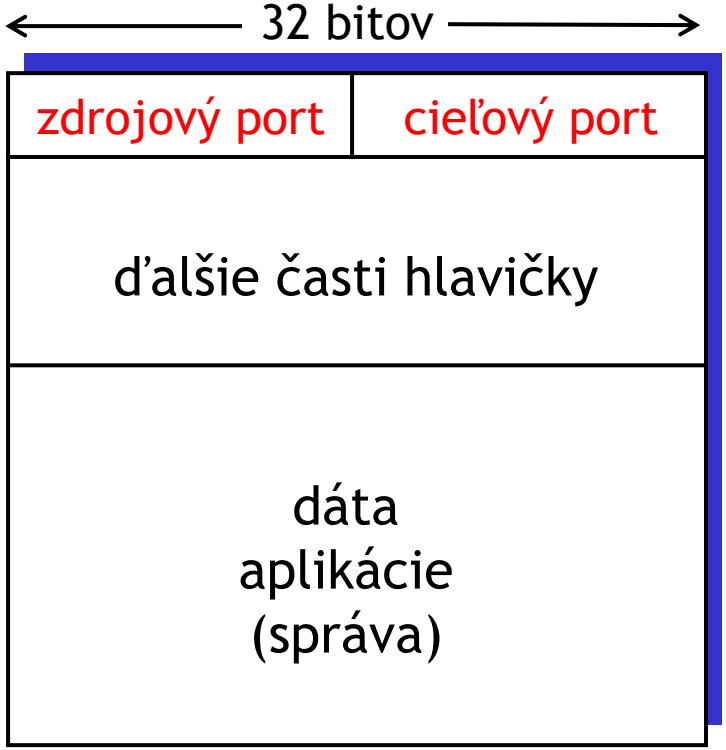
port odosielateľa a
port príjemcu

IP adresa odosielateľa a
IP adresa príjemcu

MAC adresa odosielateľa a
MAC adresa príjemcu

Odosielanie

- stanica dostane IP datagram
  - hlavička IP datagramu má zdrojovú IP adresu a cieľovú IP adresu
  - sieťová vrstva odovzdá z datagramu vyextrahovaný segment transportnej vrstve
  - hlavička segmentu má zdrojový port a cieľový port
- Transportná vrstva používa IP adresy a čísla portov na nasmerovanie segmentu k príslušnému soketu



formát segmentu TCP/UDP

Adresácia soketov v UDP

- nový soket sa asociuje s číslom portu a prípadne aj s niektorou konkrétnou IP adresou stanice:

```
soket1 = new
  DatagramSocket(12534);
soket2 = new
  DatagramSocket(12535,
              "158.197.31.4");
```

- UDP soket je jednoznačne identifikovaný dvojicou:

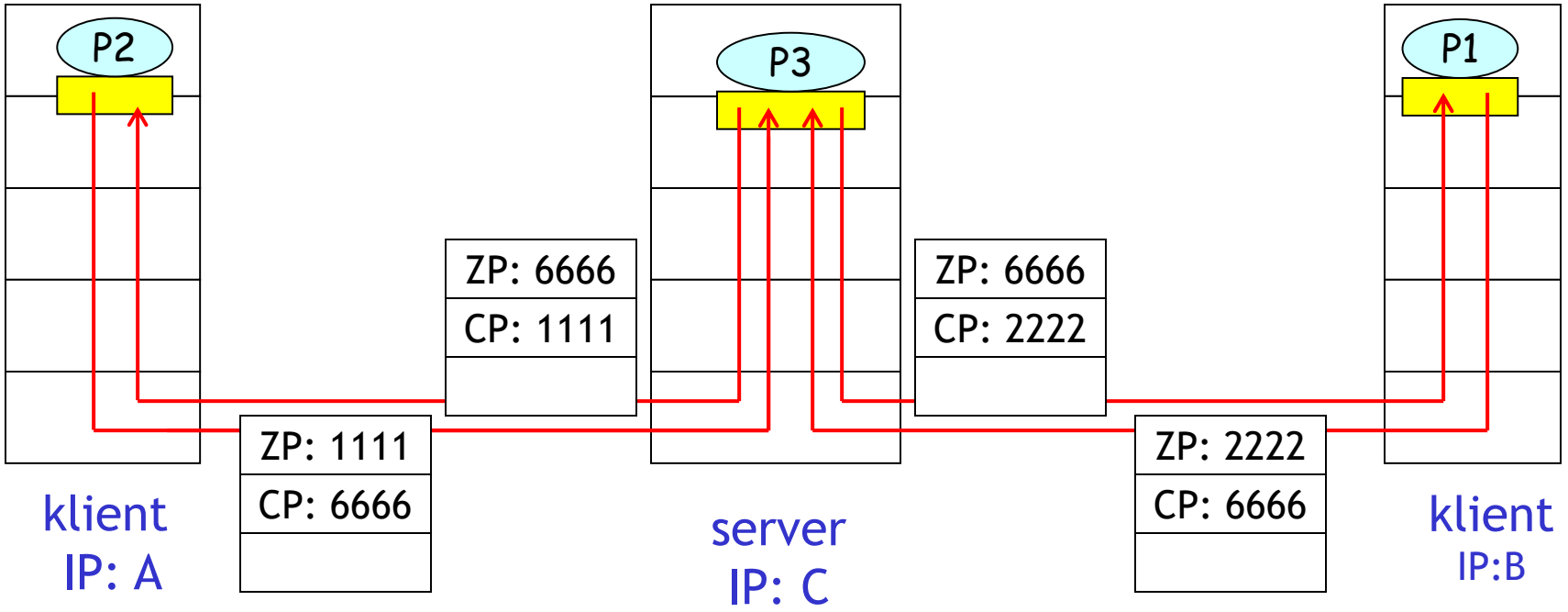
  (cieľová IP adresa, cieľový port)

- Keď stanica prijme UDP segment:
  - prečíta si číslo cieľového portu a cieľovú IP adresu
  - pošle UDP segment soketu asociovanému s týmto číslom portu
- pakety s rôznymi zdrojovými IP adresami a/alebo zdrojovými číslami portov môžu byť nasmerované k rovnakému soketu

Adresácia soketov v UDP

= soket = proces

```
DatagramSocket serverSocket = new DatagramSocket(6666);
```



Zdrojový port poskytuje "návratovú adresu"

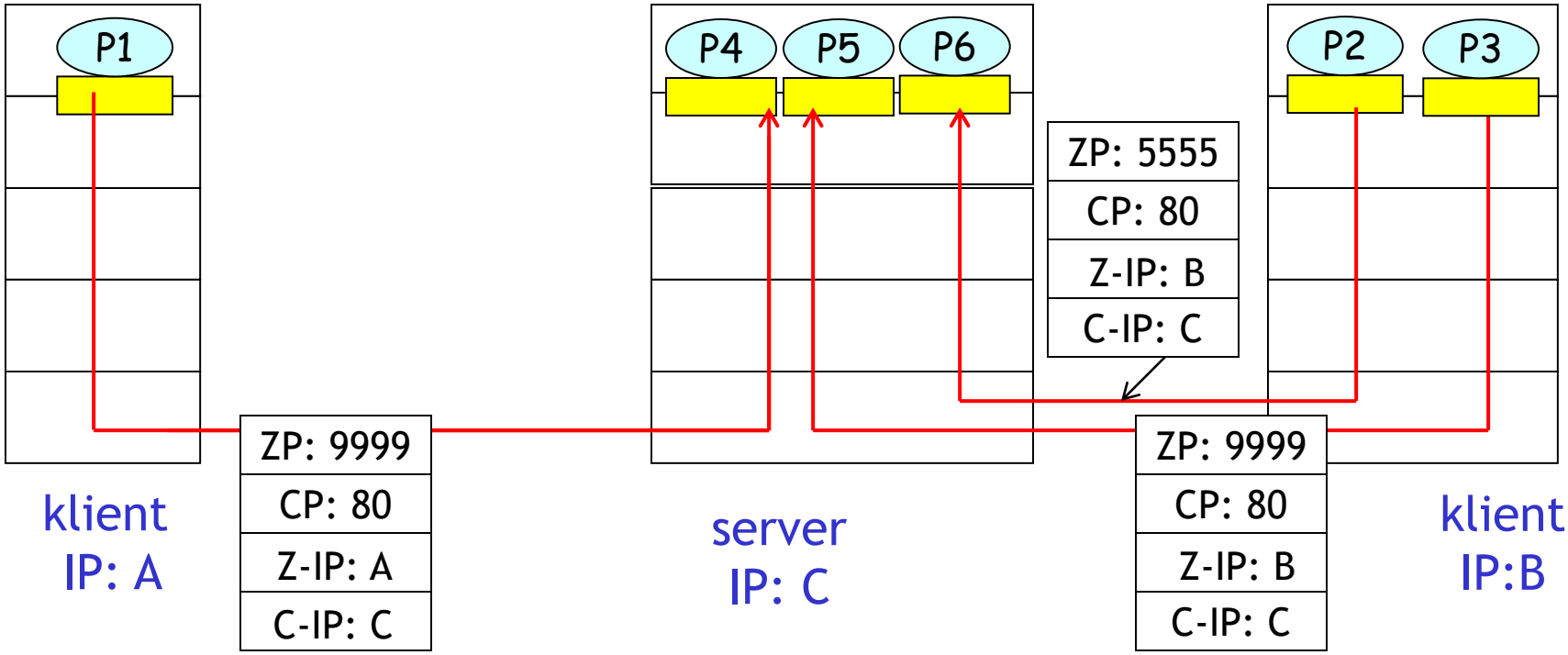
ZP = zdrojový port
CP = cieľový port

Adresácia soketov v TCP

- TCP soket je jednoznačne identifikovaný štvoricou:
  - zdrojová IP adresa
  - zdrojový port
  - cieľová IP adresa
  - cieľový port
- príjemca použije všetky štyri hodnoty na nasmerovanie k správnemu soketu

- proces môže mať súčasne otvorených veľa TCP soketov:
  - každý soket je identifikovaný svojou štvoricou
- napr. webové servery majú pre každého napojeného klienta samostatný soket
  - všetci komunikujú s rovnakým portom

Adresácia soketov v TCP



= soket = proces

ZP = zdrojový port
CP = cieľový port
Z-IP = zdrojová IP adresa
C-IP = cieľová IP adresa

UDP: User Datagram Protocol [RFC 768]

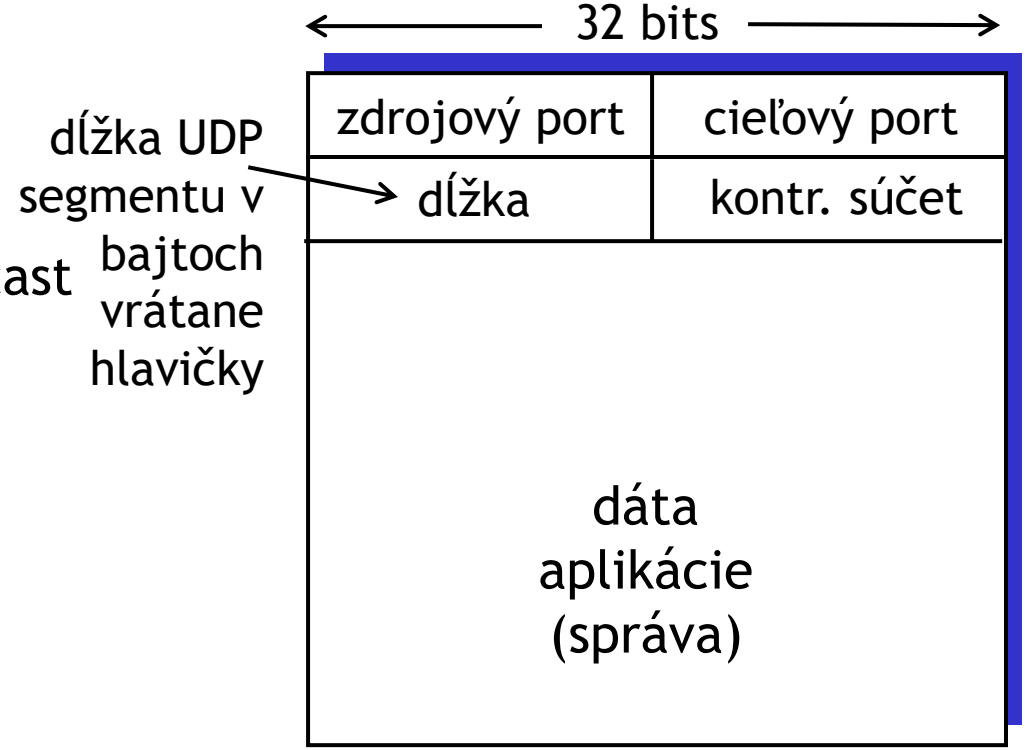
- UDP segmenty sa môžu:
  - stratiť
  - byť doručené v inom poradí, ako boli odoslané
- odosielanie „najväčším úsilím“ (best effort)
- *bezspojový protokol:*
  - žiadne nadväzovanie spojenia (handshaking) medzi odosielateľom a príjemcom
  - každý UDP segment je obslúžený nezávisle na ostatných

Načo je dobré UDP?

- žiadne inicializácie spojenia, ktoré môžu spôsobiť oneskorenie
- jednoduchý: nie je potrebné žiadne uchovávanie stavu odosielania/prijímania
- real-time prenos
- žiadna kontrola zahltenia: UDP odosiela dáta hneď, keď ich dostane z aplikačnej vrstvy
- odosielanie segmentu viacerým cieľovým staniciam (broadcast, multicast)

UDP: použitie

- Streamovanie multimédií
  - tolerujú stratu častí dát
  - závislé na rýchlosti odosielania
  - viacerým cieľom cez multicast
- DNS
- SNMP – simple network management protocol
- spoľahlivý prenos cez UDP: spoľahlivosť je riešená v aplikačnej vrstve
  - špecifické zotavovanie z chýb pre danú aplikáciu



formát UDP segmentu

Kontrolný súčet UDP segmentu

Cieľ: odhalenie "chýb" (napr. zmenené bity) v dôjdenom segmente

Odosielateľ:

- prechádza obsah segmentu ako množinu 16-bitových čísiel
- kontrolný súčet: sčítanie týchto čísiel a na záver vyrobenie inverzného čísla z výsledku sčítania (nuly vymení za jednotky a jednotky za nuly)
- odosielateľ pridá kontrolný súčet do UDP hlavičky

Príjemca:

- vypočíta súčet obsahu segmentu ako množiny 16-bitových čísiel
- nakoniec sčíta výsledok s kontrolným súčtom z UDP hlavičky.
  - Ak výsledkom nie je číslo so samými jednotkami, našli sme chybu a segment zahodíme
  - Ak áno, nenašli sme chybu
  - (ale chyba mohla aj tak nastať)

Príklad kontrolného súčtu

- Obsah segmentu tvoria červené čísla

Odosielateľ:

	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
súčet	1	1	0	0	1	0	1	0	1	1	0	0	1	0	1	0
kontrolný súčet	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1

Príjemca – spraví rovnaký súčet obsahu segmentu:

súčet →	1	1	0	0	1	0	1	0	1	1	0	0	1	0	1	0
	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1
kontrolný súčet z hlavičky →	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

samé 1 - OK

Ďakujem za pozornosť

Modifikované slajdy z knihy:

Computer Networking: A Top Down Approach ,
4th edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2007.