Poznámky ze semináře programování

Obsah

1. Hrátky v sítích

V této době už bez internetu nikdo z nás nedá ani ránu. Skoro vše, co dnes děláme na počítačích, vyžaduje připojení k internetu. Jak ale vypadá cesta od chvíle, co do prohlížeče zadáme adresu stránky? Jakou dálku překonají data, která posíláme z našeho počítače a co se s nimi kde děje?

Tato kapitola vás nejdřív provede tím, co se děje s doménou, jak se z ní stane IP adresa a jak se všechna data zabalí do paketu. Zbyde samozřejmě i čas na to se podívat jak se balíček jedniček a nul dostane do cílové destinace.

1.1. Webové adresy

Ačkoli jsou webové adresy (ve většině případů) výborně zapamatovatelné, pro přenos informací po síti mají minimální význam. Ne nadarmo se jim říká *doménová **jména***. Slouží lidem a dovolují jim napsat `google.com` místo `142.251.37.110`. Pohodlné, že?

Tomu dlouhému číslu se třemi tečkami říkáme *IP adresa*. Skládá se ze 4 bytů, které zapisujeme v desítkové soustavě a oddělujeme tečkami. Za použití jednoduché matematiky můžeme zjistit, že jich může být až 2^{32} (což je skoro 4,3 miliardy). Zdá se, že to je hodně, ale jak za chvíli zjistíme, s nedostatkem IP adres máme na světě velké problémy.

> **! Důležité**
>
> IP adresa jednoznačně identifikuje počítač připojený k nějaké počítačové síti, který využívá IP protokol.

Kdybyste si ale zjistili adresu vašeho počítače u vás doma a uděali to samé s počítačem vašeho kamaráda, může se velmi jednoduše stát, že budou mít adresu **stejnou**.

Jak je to možné? Nemá být IP adresa přeci unikátní? Ona ale je! Je unikátní ve vaší domácí síti. Je na ní váš počítač, telefon, tiskárna, lednička… a všechna tato zařízení mají odlišnou adresu. Internet je sice jedna velká síť, ale je sestavená ze spousty malých sítí. Jak ale poznáme, kde začíná jedna síť a druhá končí? Jednoduše – dělí je síťové zařízení zvané **router** (česky směrovač).

> **i Je váš router opravdu router?**
>
> Většina z vás si pod slovem router vybaví věc, ke které se připojíte telefonem přes WiFi. Ve skutečnosti jste byli oklamáni! Routery pro běžnou populaci jsou kombinací více síťových prvků (aby si ho sítěmi nepolíbený smrtelník mohl jednoduše zapojit a sledovat videa). Zaprvé obsahují tzv. *access point*, což je *ten* síťový prvek, kam se připojujete mobilem. Poté obsahují občas i *switch* (česky přepínač). To je taková prodlužovačka pro síťové kabely. A poté samozřejmě i samotný router.

Jak tedy z doménového jména získáme IP adresu? To nám zajistí protokol DNS (Domain Name System).

Doménová jména

Domény mají určitý formát, který je třeba znát před tím, než se pustíme do experimentováním s DNS. Mějme doménu `obchod.pocitace.cz`.

Skládá se ze tří částí oddělených tečkou. Části na konci se říká *doména nejvyššího řádu* (v tomto případě `cz`). Rozlišujeme dva typy – *obecné* (gTLD, generic top level domain) a *národní* (ccTLD, country code top level domain). Národní jsou třeba `.cz` nebo `.sk`, obecné jsou `.com`, `.org`, `.net` a spousty dalších.

Poté následuje *doménové jméno*. To je ta část domény, kterou si určíme při registraci. Tady je to `pocitace`.

Za doménovým jménem se nachází *subdomény*. Těch může být dokonce více než jedna! Můžeme mít tedy webovou adresu `uplne.nejlepsi.obchod.pocitace.cz`. Subdomény se již nekupují a můžeme je nastavovat libovolně. V momentě, kdy vlastníme `pocitace.cz`, můžeme vytvářet libovolné domény končící na `pocitace.cz`.

> **i WWW**
>
> Po světě panuje názor, že správný formát internetové domény je `www.*.tld`, kde `*` je cokoliv a `tld` je vybraná doména nejvyššího řádu. Ve skutečnosti tomu tak není a nejde o nic jiného než o zvyk (a o nastavení příslušných záznamů). Je však vhodné podporovat oba dva způsoby – lidé, co jsou zvyklí, psát před každou adresu `www` mohou být zmatení, když zjistí, že jim prohlížeč vyhodil chybu.

DNS

Protokol DNS je starý jako internet sám. Už v dobách ARPANETu se přemýšlelo o tom, jak lidem (americké vládě) zpříjemnit používání tehdejšího intranetu. Jako takový se protokol zformoval v osmdesátých letech.

Vše začíná u člověka, co si koupí doménu. Ano, domény se *kupují* (respektive pronajímají) a jejich ceny mohou být až astronomické (většinou tomu tak je u krátkých domén). Doménu můžete koupit u *doménového registrátora*, což je firma, která má povolení od správců domény tuto doménu prodávat. V Česku je správcem domény `.cz` společnost CZ.NIC. Doménovým registrátorem naší národní domény je třeba Wedos.cz, ale i spousty dalších.

Osoba, která si doménu koupí, k ní může přidávat tzv. *DNS záznamy*. Mají několik typů a určují to, na co se bude doména překládat. Nutno dodat, že DNS nezajišťuje jen překlad domén na IP adresy. Umí toho mnohem víc (bez DNS by nefungoval e-mail tak, jak ho známe).

Nejdůležitější záznam, který budeme používat, je `A` záznam. `A` znamená *address record* a umožňuje nám přesně to, co chceme. Představme si, že vlastníme doménu `pocitace.cz`. Provozujeme obchod s počítači a potřebujeme, aby byl dostupný na adrese `obchod.pocitace.cz`. Tento obchod máme na nějakém serveru, který má IP adresu `145.33.63.244`. Vytvoříme tedy `A` záznam, který bude vypadat následovně:

```
obchod  IN  A  145.33.63.244
```

První část určuje, jaká poddoména se má přeložit. Druhá část, slovíčko `IN`, je zkratka pro „INternet“. Následuje typ záznamu a IP adresa, na kterou se má daná doména přeložit.

Záznamy se uchovávají na *name serverech*, které rozdělujeme do dvou druhů. *Autoritativní* nameservery mají na starosti samotné ukládání záznamů a jsou poslední *autoritou* v otázce „je tento záznam opravdu platný?“ Dalším druhem jsou servery *rekurzivní*, které cachují záznamy, aby na autoritativní servery nebyl moc velký nápor. Jsou to právě rekurzivní servery, kterých se ptají naše počítače.

> **TTL**
>
> Údaj o „životnosti“ je v DNS důležitý. Můžeme se ptát, jestli je záznam *stále* platný. Subjekt, co doménu vlastní, si může záznamy libovolně měnit a kdybychom neměli mechanismus na obnovování cachovaných dat, měli bychom stále stejný (starý) údaj. K tomu slouží TTL (z anglického Time To Live). Každý záznam má takový údaj, který říká, za jakou dobu se máme znovu autoritativních serverů zeptat, jestli se záznam nezměnil. Tuto práci za nás dělají rekurzivní servery – dotaz na aktualizaci cachovaných záznamů jde tedy od rekurzivních serverů k autoritativním, nikoli naopak.

První DNS cache je hned na vašem počítači (na Linuxu se o ni stará démon `systemd-resolved`, na Windows ji můžete zobrazit pomocí příkazu `ipconfig`). Ta další je hned nedaleko, a to ve vašem routeru.

Příklad překladu doménového jména se známým rekurzivním serverem

Představte si, že jste počítač – dobrá představa, že? Uživatel se chce podívat na webovou stránku s webovou adresou `d3s.mff.cuni.cz`. Webový prohlížeč nás zaúkoloval, abychom našli IP adresu, která je s touto doménou spojená.

Budeme používat program `dig`, který nám umožní nahlédnout do záznamů name serverů.

Jednodušší (a mnohem častější) varianta je, když máme v nastavení připojení k internetu (typicky v nastavení vašeho počítače) vyplněný DNS server, kterého se máme dotazovat. Většinou to je rekurzivní server vašeho poskytovatele internetu nebo nějaký velký rekurzivní server od firem jako Cloudflare° nebo Google°. Existují i více privacy-friendly možnosti, jako Quad9°.

Tyto velké servery hromadně cachují záznamy a tím zaručují vysokou pravděpodovnost, že náš požadavek u sebe budou mít. Nemusí se tedy dotazovat autoritativního serveru, což může trvat dlouho.

IP adresa rekurzivního serveru Quad9 je (nepřekvapivě) `9.9.9.9`. Pojďme se ho zeptat, jestli nemůže pomoct s naším úkolem.

```
$ dig d3s.mff.cuni.cz @9.9.9.9
```
Shell

```
; <<>> DiG 9.20.2 <<>> d3s.mff.cuni.cz @9.9.9.9
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53181
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;d3s.mff.cuni.cz.               IN      A

;; ANSWER SECTION:
d3s.mff.cuni.cz.        28792   IN      A       195.113.20.60

;; Query time: 33 msec
;; SERVER: 9.9.9.9#53(9.9.9.9) (UDP)
;; WHEN: Fri Oct 18 12:08:14 CEST 2024
;; MSG SIZE  rcvd: 60
```

Formát, který `dig` vrací, je poněkud zmatečný, ale to zvládneme. Nejdřív by bylo hodno zjistit, jestli jsme vůbec dostali odpověď. Řádek 5 nám říká, že jsme poslali jeden dotaz (`QUERY: 1`) a dostali jednu odpověď (`ANSWER: 1`). To zní dobře!

Na řádku 10 je zformulován náš dotaz. Přeloženo do češtiny by to bylo „Máš u sebe nějaký `A` záznam pro doménu `d3s.mff.cuni.cz` ?“

Na řádku 13 vidíme odpověď. IP adresa `195.113.20.60` opravdu odpovídá požadované doméně. Číslo `28792` , které je také v odpovědi, není nic jiného, než TTL. Pokud spustíte stejný `dig` příkaz víckrat po sobě, zjistíte, že se číslo bude snižovat.

Příklad překladu neznámého doménového jména

Tato situace nastává velmi zřídka, obvykle v těchto případech:

- Rekurzivní server nikdy doménu neviděl (myšleno **celou** doménu, tedy i `.cz`)
- Vypršel údaj TTL a rekurzivní server neví, koho se zeptat
- Jsme zvídaví a chceme se podívat, jak vypadá struktura DNS serverů
- Jsme síťový administrátor a řešíme problém

Když nevíme, obracíme se k bohu. V tomto případě k organizaci IANA (Internet Assigned Numbers Authority), která spravuje a dohlíží na systém DNS. Má také veřejný seznam° tzv. *kořenových nameserverů* (root servers), které jsou spravované různými organizacemi po světě a můžeme přes ně hledat *jakoukoli* doménu.

```
1   a.root-servers.net  198.41.0.4, 2001:503:ba3e::2:30   Verisign, Inc.
2   b.root-servers.net  170.247.170.2, 2801:1b8:10::b   University of Southern California,
    Information Sciences Institute
3   c.root-servers.net  192.33.4.12, 2001:500:2::c  Cogent Communications
4   d.root-servers.net  199.7.91.13, 2001:500:2d::d   University of Maryland
5   e.root-servers.net  192.203.230.10, 2001:500:a8::e  NASA (Ames Research Center)
6   f.root-servers.net  192.5.5.241, 2001:500:2f::f   Internet Systems Consortium, Inc.
7   g.root-servers.net  192.112.36.4, 2001:500:12::d0d  US Department of Defense (NIC)
8   h.root-servers.net  198.97.190.53, 2001:500:1::53   US Army (Research Lab)
9   i.root-servers.net  192.36.148.17, 2001:7fe::53   Netnod
10  j.root-servers.net  192.58.128.30, 2001:503:c27::2:30   Verisign, Inc.
11  k.root-servers.net  193.0.14.129, 2001:7fd::1   RIPE NCC
12  l.root-servers.net  199.7.83.42, 2001:500:9f::42  ICANN
13  m.root-servers.net  202.12.27.33, 2001:dc3::35  WIDE Project
```

Zkusme přeložit doménu `d3s.mff.cuni.cz` pomocí kořenových nameserverů. Zeptejme se tedy hned toho prvního s IP adresou `198.41.0.4` .

```
$ dig d3s.mff.cuni.cz @198.41.0.4                                    Shell
```

```
1   ; <<>> DiG 9.20.2 <<>> d3s.mff.cuni.cz @198.41.0.4
2   ;; global options: +cmd
3   ;; Got answer:
4   ;; ⟶»HEADER«⟵ opcode: QUERY, status: NOERROR, id: 5294
5   ;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 9
6   ;; WARNING: recursion requested but not available
7
8   ;; OPT PSEUDOSECTION:
9   ; EDNS: version: 0, flags:; udp: 4096
10  ;; QUESTION SECTION:
```

```
11 ;d3s.mff.cuni.cz.               IN      A
12
13 ;; AUTHORITY SECTION:
14 cz.                      172800  IN      NS      a.ns.nic.cz.
15 cz.                      172800  IN      NS      c.ns.nic.cz.
16 cz.                      172800  IN      NS      b.ns.nic.cz.
17 cz.                      172800  IN      NS      d.ns.nic.cz.
18
19 ;; ADDITIONAL SECTION:
20 a.ns.nic.cz.             172800  IN      A       194.0.12.1
21 a.ns.nic.cz.             172800  IN      AAAA    2001:678:f::1
22 c.ns.nic.cz.             172800  IN      A       194.0.14.1
23 c.ns.nic.cz.             172800  IN      AAAA    2001:678:11::1
24 b.ns.nic.cz.             172800  IN      A       194.0.13.1
25 b.ns.nic.cz.             172800  IN      AAAA    2001:678:10::1
26 d.ns.nic.cz.             172800  IN      A       193.29.206.1
27 d.ns.nic.cz.             172800  IN      AAAA    2001:678:1::1
28
29 ;; Query time: 106 msec
30 ;; SERVER: 198.41.0.4#53(198.41.0.4) (UDP)
31 ;; WHEN: Fri Oct 18 12:41:40 CEST 2024
32 ;; MSG SIZE  rcvd: 291
```

Řádek 5 nám říká, že jsme poslali jeden dotaz, ale nedostali žádnou odpověď. To je divné. Není to ale úplně pravda. Dostali jsme odpověď, která nám říká „Nevím, kde je `d3s.mff.cuni.cz`, ale vím o *autoritě*, která by něco vědět mohla“.

Na řádcích 14 až 17 nám kořenový nameserver vypsal čtyři nameservery, které spravují národní doménu `cz`. Byl i tak hodný, že nám rovnou řekl jejich IP adresy (řádky 20 až 27).

Zeptáme se tedy autoritativního nameserveru `a.ns.nic.cz`, který má IP adresu `194.0.12.1`, jestli neví, kde je `d3s.mff.cuni.cz`.

```
$ dig d3s.mff.cuni.cz @194.0.12.1                                    Shell
```

```
1  ; <<>> DiG 9.20.2 <<>> d3s.mff.cuni.cz @194.0.12.1
2  ;; global options: +cmd
3  ;; Got answer:
4  ;; ⟶»HEADER«⟵ opcode: QUERY, status: NOERROR, id: 26046
5  ;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 7
6  ;; WARNING: recursion requested but not available
7
8  ;; OPT PSEUDOSECTION:
9  ; EDNS: version: 0, flags:; udp: 1232
10 ;; QUESTION SECTION:
11 ;d3s.mff.cuni.cz.               IN      A
12
13 ;; AUTHORITY SECTION:
14 cuni.cz.                3600    IN      NS      nsa.ces.net.
15 cuni.cz.                3600    IN      NS      cuce.ruk.cuni.cz.
16 cuni.cz.                3600    IN      NS      david.ruk.cuni.cz.
17 cuni.cz.                3600    IN      NS      golias.ruk.cuni.cz.
18
19 ;; ADDITIONAL SECTION:
```

```
20 cuce.ruk.cuni.cz.       3600    IN      A       195.113.0.8
21 cuce.ruk.cuni.cz.       3600    IN      AAAA    2001:718:1e03:1::8
22 david.ruk.cuni.cz.      3600    IN      A       78.128.213.242
23 david.ruk.cuni.cz.      3600    IN      AAAA    2001:718:1203:502::2
24 golias.ruk.cuni.cz.     3600    IN      A       195.113.0.2
25 golias.ruk.cuni.cz.     3600    IN      AAAA    2001:718:1e03:1::2
26
27 ;; Query time: 133 msec
28 ;; SERVER: 194.0.12.1#53(194.0.12.1) (UDP)
29 ;; WHEN: Fri Oct 18 15:40:55 CEST 2024
30 ;; MSG SIZE  rcvd: 272
```

Dostali jsme podobnou odpověď jako předtím – `a.ns.nic.cz` neví, kde `d3s.mff.cuni.cz` je, ale ví to některý z nameserverů, které nám poslal v `AUTHORITY SECTION`.

Zeptejme se serveru `david.ruk.cuni.cz`.

```
$ dig d3s.mff.cuni.cz @78.128.213.242
```
Shell

```
1  ; <<>> DiG 9.20.2 <<>> d3s.mff.cuni.cz @78.128.213.242
2  ;; global options: +cmd
3  ;; Got answer:
4  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13150
5  ;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 3
6  ;; WARNING: recursion requested but not available
7
8  ;; OPT PSEUDOSECTION:
9  ; EDNS: version: 0, flags:; udp: 1232
10 ; COOKIE: 9d3b528254246b9301000000671266 3095eeca8b8db7a9cc (good)
11 ;; QUESTION SECTION:
12 ;d3s.mff.cuni.cz.               IN      A
13
14 ;; AUTHORITY SECTION:
15 d3s.mff.cuni.cz.        28800   IN      NS      sns.ms.mff.cuni.cz.
16 d3s.mff.cuni.cz.        28800   IN      NS      ns.ms.mff.cuni.cz.
17
18 ;; ADDITIONAL SECTION:
19 sns.ms.mff.cuni.cz.     28800   IN      A       195.113.20.77
20 ns.ms.mff.cuni.cz.      28800   IN      A       195.113.20.71
21
22 ;; Query time: 53 msec
23 ;; SERVER: 78.128.213.242#53(78.128.213.242) (UDP)
24 ;; WHEN: Fri Oct 18 15:44:15 CEST 2024
25 ;; MSG SIZE  rcvd: 142
```

Pokračuje stejná hra na přehazovanou, zeptejme se serveru `ns.ms.mff.cuni.cz`.

> **Primární a sekundární nameservery**
>
> Síť DNS serverů je distribuovaný systém. Informace o té samé doméně se mohou nacházet na více místech najednou. Pokud spadne jeden rekurzivní server, žádné velké drama se neděje. Problém ovšem je, když je nedostupný autoritativní server. V ten moment nemůžeme od nikoho dostat platné záznamy. Proto se vždy DNS záznamy ukládají na více míst najednou. Nejběžnější je mít primární a sekundární nameserver, kde jsou uložená stejná data. U kritických systému jich může být samozřejmě více.
>
> V tomto případě se záznamy o doméně `d3s.mff.cuni.cz` uchovávají na dvou nameserverech. Jeden je `ns.ms.mff.cuni.cz`, druhý `sns.ms.mff.cuni.cz`. Prefix `s` u druhého z nich nejspíše znamená „sekundární“.

```shell
$ dig d3s.mff.cuni.cz @195.113.20.77
```

```
; <<>> DiG 9.20.2 <<>> d3s.mff.cuni.cz @195.113.20.77
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64478
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 27540d911313ad2f010000006712668ebac58a2f34749395 (good)
;; QUESTION SECTION:
;d3s.mff.cuni.cz.               IN      A

;; ANSWER SECTION:
d3s.mff.cuni.cz.        28800   IN      A       195.113.20.60

;; Query time: 166 msec
;; SERVER: 195.113.20.77#53(195.113.20.77) (UDP)
;; WHEN: Fri Oct 18 15:45:49 CEST 2024
;; MSG SIZE  rcvd: 88
```

Konečně jsme se dočkali odpovědi! Webová stránka, na kterou ukazuje doména `d3s.mff.cuni.cz` je na IP adrese `195.113.20.60`. Vše to, co jsme tu prováděli ručně, by musel provést rekurzivní DNS server.

1.2. Protokol HTTP

Když jsme se dostali k IP adrese počítače (serveru), od kterého požadujeme zaslání dat, které se nám na počítači vykreslí jako krásná webová stránka, můžeme pokračovat zasláním HTTP *požadavku* (HTTP request).

Ten pošleme pomocí stejnojmenného protokolu. HTTP je *klient-server* protokol, kde *klient* navazuje spojení se *serverem*. Klient posílá požadavky, server na ně odpovídá.

Pro demonstraci budeme používat verzi protokolu HTTP 1.1°. V dnešní době už se nepoužívá a byl plně nahrazen verzí 2. Velká část komunikace už dnes používá HTTP verzi 3. Důvod volby verze 1.1 je jednoduchý – tato verze je totiž posílaná jako *plaintext* (čitelný text). Ostatní využívají serializace do binárního formátu a jsou tak hůře čitelné.

V dnešní době už je těžké najít webový server, který podporuje HTTP 1.1. Naštěstí existují testovací webové stránky, které ho v rámci experimentů umožňují používat. Jednou z takových stránek je httpstat.us°.

Použijeme program `telnet`, který naváže spojení s HTTP serverem a my s ním budeme moci komunikovat. Telnet je předchůdce SSH, má ale jednu velkou nevýhodu. Je to **nešifrovaný** protokol, tudíž bychom ním neměli posílat nic citlivého. Kdo by se tomu také divil, když je s námi přes 50 let.

```shell
$ telnet httpstat.us 80
```

Shell

Tímto příkazem říkáme, že chceme navázat spojení se serverem na adrese `httpstat.us` a s programem, který běží na portu 80. O *portech* si řekneme něco později, prozatím nám bude stačit analogie k IP adresám. IP adresy identifikují počítače na síti, porty identifikují jednotlivé programy na počítači, které jsou schopné komunikovat pomocí síťových protokolů.

HTTP má tradičně přiřazený port `80` (telnet má `23`, SSH najdeme na portu `22`). Číslo portu můžeme měnit, ale většinou to je považované za antivzor.

Po zadání příkazu výše se dostaneme do prostředí Telnetu. Teď můžeme zadávat cokoli, co chceme poslat pomocí HTTP na server.

```
Trying 20.40.202.3...
Connected to httpstat.us.
Escape character is '^]'.

```

Napíšeme tedy následující a stiskneme dvakrát klávesu Enter.

```
GET / HTTP/1.1
Host: httpstat.us
```

Tento požadavek říká, že chceme získat stránku, která je na *cestě* (angl. *path*) `/` a chceme použít verzi `HTTP/1.1`. Druhý řádek obsahuje *hlavičku* – není to nic jiného, než dodatečná informace ve formátu `klíč: hodnota`. Hlavička `Host` je ve verzi HTTP 1.1 povinná, protože dovoluje serveru, který požadavek zpracovává, rozhodnout se, jaká data má poslat zpět (HTTP 1.1 totiž dovoluje poskytovat více různých webových stránek z jednoho serveru).

Odpověď od serveru vypadá takto:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Date: Fri, 18 Oct 2024 15:17:38 GMT
Server: Kestrel
Set-Cookie: ARRAffinity=4084694ccb4e33182ae90ba038dfb1acd4517551657c81596510cc62148e7f83;Pa
Transfer-Encoding: chunked
Request-Context: appId=cid-v1:3548b0f5-7f75-492f-82bb-b6eb0e864e53

a2e

<!DOCTYPE html>
<html lang="en">
<head>
    <title>httpstat.us</title>
    <link href="/css/main.css" rel="stylesheet" type="text/css" />
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
17  </head>
18  <body>
19      <div id="content">
20          <header>
21              <h1 id="title">httpstat.us</h1>
22  ...
```

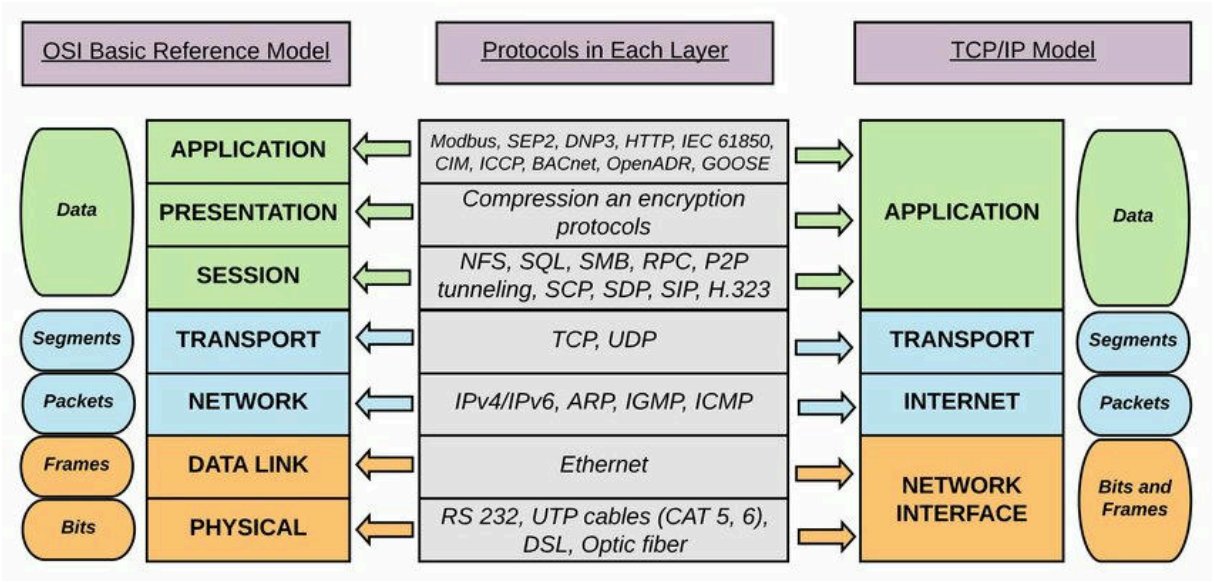
Je rozdělená do dvou částí – hlavičky a *těla*. Na prvním řádku se nachází důležitá informace. Za verzí protokolu je napsáno `200 OK`. Toto číslo s popiskem se jmenuje *status kód* a říká nám, jestli náš požadavek uspěl nebo ne. HTTP server může tyto status kódy nastavovat jak chce a odlišovat tak různé druhy chyb a úspěchů. Pro více informací se podívejte sem°. Určitě vám ale všem něco říká `404 Not Found`...

Jako další vidíme pár hlaviček. Hlavička `Content-Type` určuje typ obsahu (ten se nejen v HTTP určuje tzv. *MIME typem*), co nám server posílá a jeho kódování (zde UTF-8). `Date` hlavička říká, kdy byla odpověď odeslána. Dalšími hlavičkami se zaobírat nebudeme.

Třešnička na dortu odpovědi je samotný zdrojový kód webové stránky. Ten je psaný v HTML. Pokud HTTP požadavek zadává prohlížeč, ten odpověď zpracuje a vykreslí.

1.3. OSI a TCP/IP model

Ve snaze standardizovat strukturu počítačových sítí byl organizací ISO (International Organization for Standards) koncept zvaný *OSI model*. Je to teoretický model, kterým by se *měly* řídit všechny částí sítě. Má sedm vrstev, přičemž platí, že vyšší vrstva využívá prostředky vrstvy, kterou má pod sebou. Tento teoretický výmysl se zhmotnil do podoby TCP/IP modelu – ten využívá jen čtyři vrstvy (respektive, tři vrstvy z OSI modelu sloučil do jedné).



Obrázek 1: Porovnání modelu OSI a TCP/IP a protokolů, které se na nich využívají

Na každé vrstvě se využívají jiné protokoly – ve skutečnosti je jejich souhra právě to, co umožnuje efektivní přenos dat po síti. Na každé vrstrvě se data z té předchozí (té nad ní) „zabalí“ do nových dat, které vytvoří vrstva o úroveň níž. To samozřejmě funguje i naopak, data se mohou i „rozbalit“. Je důležité říct, že ne všechny síťové prvky implementují všechny vrstvy – třeba směrovače data transportní vrstvy vůbec nezajímají.

Aplikační vrstva

O protokolech, které pracují na aplikační vrstvě, jsme si již říkali – HTTP, DNS, Telnet a SSH. Aplikační vrstva má za úkol umožnit koncovým aplikacím komunikovat přes síť.

Transportní vrstva

Zde už se začínají dít zajímavé věci. Transportní vrstva má za úkol zajistit end-to-end komunikaci mezi dvěma počítači. Její protokoly mohou přidávat různé užitečné vlastnosti, které bychom od přenosu chtěli – snahu o koreknost a bezchybnost dat, opravu chyb a navazování a udržování spojení.

Dva hlavní protokoly, které se na transportní vrstvě běžně používají jsou TCP (Transmission Control Protocol) a UDP (User segment Protocol).

UDP

Začneme tím jednodušším protokolem. UDP se používá v případech kdy nám nezáleží na tom, jestli od příjemce požadujeme potvrzení o přijetí dat. Protokol UDP se mnohdy popisuje jako *nespolehlivý*, lepší je ale říct, že neposkytuje *záruku doručení*. Je také *connectionless*, tedy se před posíláním dat nenavazuje žádné spojení. Odesílatel „háže“ po síti data na příjemce a ten se je snaží všechna pochytat.

Co se korekce dat týče, tak UDP žádnou takovou funkcionalitu nemá. Disponuje pouze primitivním *kontrolním součtem* na ověření integrity dat.

> **i Kontrolní součet (checksum)**
>
> Když přijmeme data, tak bychom chtěli jednoduše zjistit, jestli jsou úplná. Tedy jestli nedošlo během přenosu k jejich pozměnění (většinou neúmyslnému). K tomu slouží kontrolní součet. Na data se pošle určitá operace, jejímž výstupem je právě kontrolní součet. Tato operace by měla být rychlá, aby se kontrolní součet dal rychle provést jak na straně odesílatele tak na straně příjemce. Přijemce provede operaci na datech a ty společně s kontrolním součtem pošle. Příjemce provede to samé a porovná přijatý checksum s checksumem, který přišel od odesílatele. Pokud se rovnají, je vše v pořádku.
>
> Typický se pro tvorbu kontrolního součtu používají hešovací funkce, nejčastěji CRC, SHA nebo MD5. Pro primitivní implementaci však postačí i operace modulo.

Proč bychom ale takový protokol chtěli používat, když je „nespolehlivý“, nemá žádné schopnosti na opravu chyb nebo na detekci nepřijatých dat? UDP se typicky používá v případech, kdy nám na ztracených datech moc nezáleží. Třeba takové streamování videa nevyžaduje, aby se k uživateli dostal každý frame – lidské oko to není schopné rozpoznat. Dalšími aplikacemi jsou přenos hlasu po síti (VoIP) nebo různé herní servery.

Také nám již známé DNS využívá (ve většině případů) UDP na transportní vrstvě. Povaha DNS požadavků nevyžaduje trvalé spojení mezi dvěma servery, protože by se daly označit za jednorázové. V případě neúspěchu se jednoduše pošle další.

TCP

Protokol TCP je komplikovanějším bratříčkem (spíše velkým bratrem) UDP. Hlavní rozdíly jsou navazování spojení před komunikací a také „spolehlivost“. Znovu je nutné podotknout, že spolehlivostí je myšleno potvrzování přijatých dat. Umí také automaticky znovu posílat data, která byla ztracena při transportu po síti.

Vylepšením od UDP je také to, že pokud data přijdou v nesprávném pořadí (např. při rozdělení velkého bloku dat do několika menších), TCP je umí samo seřadit – používá k tomu pořádová čísla (anglicky *sequence number*), která s daty posílá. Potvrzovací čísla (anglicky *acknowledgement number*) logicky slouží k potvrzování přijatých dat.

Balíčkům dat, které se přes TCP posílají, se říká *segmenty*. Ty se skládají z hlavičky a těla. Do hlavičky se dávají údaje potřebné k přenosu dat v těle (příznaky, pořadové a potvrzovací čísla, a další).

Navázání spojení

Abychom mohli posílat data pomocí TCP, musíme nejdřív navázat spojení s protistranou. V TCP se používá trojcestný handshaking (anglicky *three-way handshake*). Trojcestný je proto, že se mezi klientem a server vymění tři zprávy, kterými si strany oznámí, že jsou schopné přijímat a posílat data.

Jak už název napovídá, trojcestný handshake má tři kroky.

1. Klient pošle serveru segment s nastaveným `SYN` příznakem a náhodně generovaným pořadovým číslem s.
2. Server pošle klientovi segment s nastavenými `SYN` a `ACK` příznaky, potvrzovacím číslem $s + 1$ a pořadovým číslem t.
3. Klient pošle serveru segment s nastaveným `ACK` příznakem, pořadovým číslem $s + 1$ a potvrzovacím číslem $t + 1$.

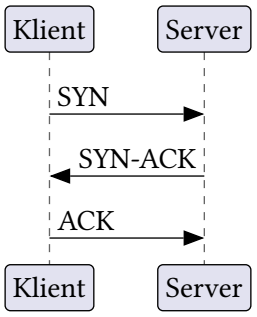


Diagram 1: Trojcestný handshake

Po dokončení trojcestného handshaku se otevře roura, kterou se můžou začít posílat data.

Výměna dat

Při navazování spojení se určily dvě důležitá čísla – s a t. Toto byla počáteční pořadová čísla, které si strany vyměnily při trojcestném handshaku.

Ukážeme si příklad, jak vypadá komunikace na úrovni TCP mezi klientem a serverem při jednoduchém HTTP dotazu z kapitoly o protokolu HTTP. Budeme posílat `GET` požadavek na stránku `httpstat.us/200`, která vrátí.

Jsme nyní ve stavu, kdy jsme navázali TCP spojení se serverem pomocí trojcestného handshaku. U každé šipky znázorňující zaslání dat protistraně budeme vypisovat údaje, které jsou důležité – pořadové číslo (v diagramu jako `seq`), potvrzovací číslo (v diagramu jako `ack`) a délku dat (`len`), které se posílají. Výměna bude vypadat následovně.

Po trojcestném handshaku bylo pořadové číslo ustanoveno na hodnotu 1883028313 a potvrzovací číslo na 1538410364. Nutno dodat, že v prvním segmentu poslaném ve trojcestném handshaku bylo pořadové číslo 1883028312, potvrzovací ve druhém segmentu bylo 1538410363.

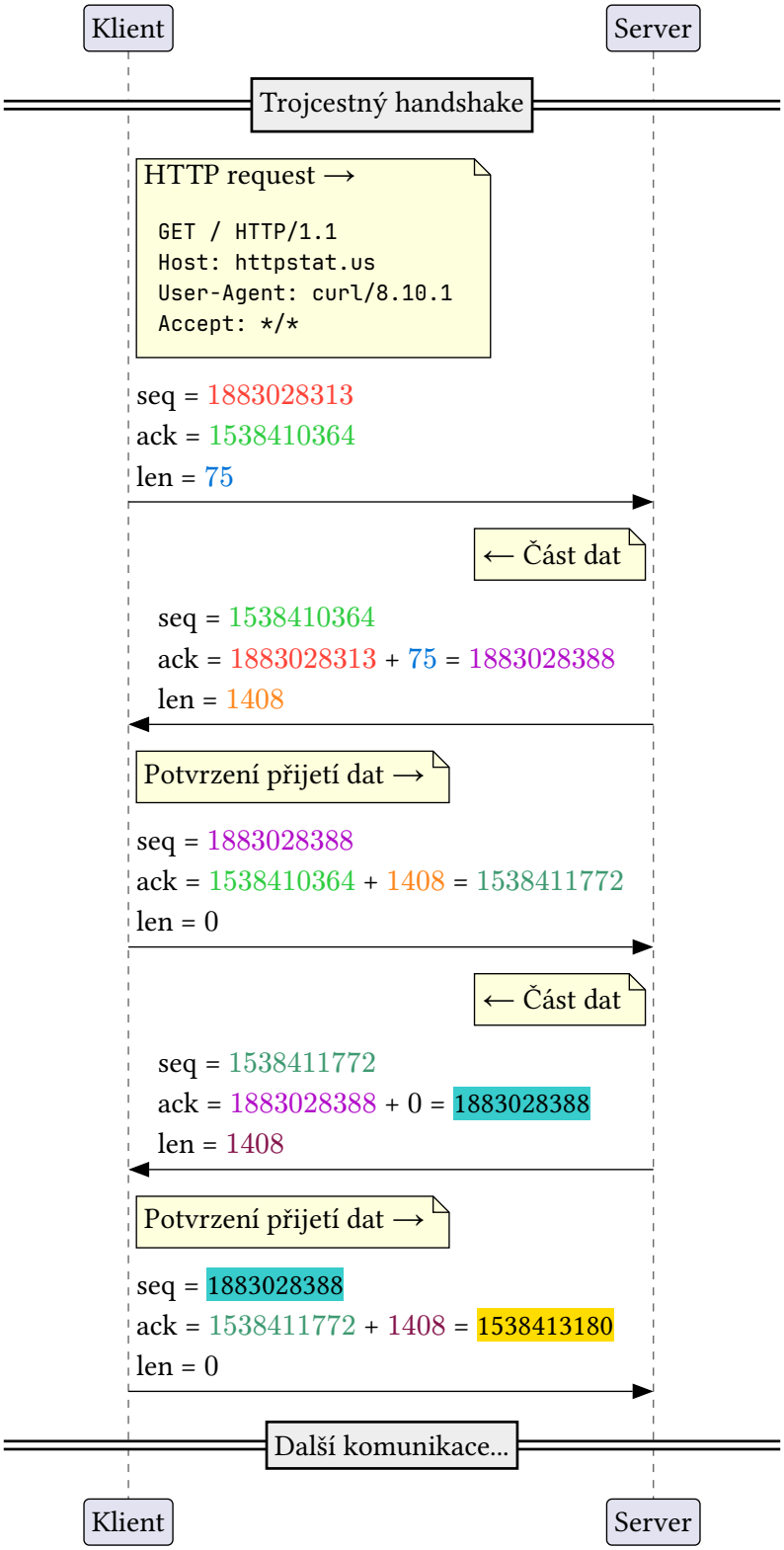


Diagram 2: Komunikace po TCP

První přenesený segment (po trojcestném handshaku) je samotný HTTP request. Text požadavku se umístí do těla TCP segmentu (jak jsme psali, data se „zabalují“ do balíčku směrem dolů). HTTP požadavek přijde od aplikační vrstvy a je zabalen vrstvou transportní (zde protokolem TCP).

Server segment zpracuje a vrátí druhý segment. Ten má stejné pořadové číslo, jako první segment. V potvrzovacím čísle je součet délky dat prvního segmentu a sekvenčního čísla prvního segmentu – protistrana tím říká, že obdržela 75 bytů dat. Takto to jde dál a dál, dokud se nepřenesou celá data.

Je pěkné si všimnout toho, co se stane, když od potvrzovacího čísla posílaného serveru odečteme počáteční potvrzovací číslo (tedy to, které bylo posláno v druhém segmentu handshaku, tedy 1538410363). Ve třetím segmentu v diagramu je potvrzovací číslo 1538411772, po odečtení vyjde $1538411772 - 1538410363 = 1409$. Nevyšlo nám náhodou o jedna více? Vždyť délka dat byla 1408. Nevyšlo! Tímto serveru v podstatě říkáme, že očekáváme další segment, jehož data začínají od tisícčtyřstéhodevátého bytu dat, které nám chce poslat.

Ukončení spojení

Jako jsme TCP spojení otevřeli, musíme ho také zavřít. Proces je podobný navazování spojení a používají se k němu segmenty s příznaky `FIN` a `ACK`. TCP spojení zavírá strana, která již nechce posílat žádná data. Bez újmy na obecnosti řekneme, že to je klient.

1. Klient pošle segment s příznakem `FIN` a pořadovým číslem u.
2. Server odpoví segmentem s příznakem `ACK`, potvrzovacím číslem $u + 1$ a pořadovým číslem v.
3. Server odešle další segment, tentokrát s příznakem `FIN` s pořadovým číslem v.
4. Klient odešle segment s příznakem `ACK` s potvrzovacím číslem $v + 1$.

Nutno dodat, že mohou nastat situace, kdy ukončovací proces může být kratší (když se obě strany rozhodnou ve stejný moment, že chtějí připojení uzavřít). Toto je ovšem standard.

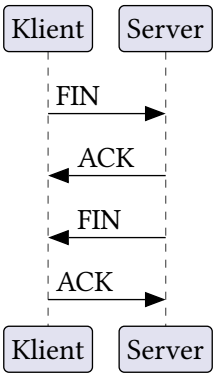


Diagram 3: Ukončení TCP spojení

Vrstva síťového rozhraní

Než se vrhneme na síťovou vrstvu je vhodné si něco říci o vrstvě síťového rozhraní. Je to nejnižší vrstva TCP/IP modelu, která se stará přímo o posílání nul a jedniček po fyzickém médiu (kroucený dvoulinka, optický kabel, a další). Aby věděla, kam je poslat, využívá svůj druh adres – MAC adresy (z angl. *Media Access Control*).

> **!** **MAC adresy**
>
> Stejně jako *porty* identifikují procesy a *IP adresy* identifikují počítače na síti, **MAC adresy identifikují přímo síťové karty v počítačích**. MAC adresa se síťové kartě dává přímo ve výrobě a měla by být unikátní v rámci celého světa (žádné dvě síťové karty nemohou mít stejnou adresu). Také se jim občas říká *fyzická adresa*, což už v této době není zas tak přesné, protože se MAC adresa dá různými způsoby měnit.

Na vrstvě síťového rozhraní se jako na ostatních vrstvách používají protokoly. Tím nejrozšířenějším je protokol *Ethernet* (ano, je to jméno protokolu, nikoli název kabelu).

> **Síťové kabely**
>
> K připojování počítačů se dá použít mnoho typů fyzického média. Tím nejčastějším jsou *kroucené dvojlinky* (anglicky *twisted pair*) s koncovkou RJ-45. Můžeme mít kabel stíněný (STP, *shielded twisted pair*) nebo nestíněný (UTP, *unshielded twisted pair*) – stíněné kabely disponují samozřejmě větší odolností proti vnějšímu rušení. Proč je to vlastně ale dvojlinka *kroucená*? Důvod je, znovu, ochrana proti elektromagnetickému záření z okolí. Zároveň kroucení kabelů zajišťuje lepší průchod dat.
>
> V běžných případech použijeme kroucené dvojlinky kategorie Cat5, Cat5e nebo Cat6 – dalo by se říci, že čím vyšší číslo kategorie, tím vyšší maximální přenosová rychlost. V jednom kabelu také není jen jedna kroucená dvojlinka, nýbrž hned čtyři (celkem tedy osm měděných drátků).

MAC adresy protokolu Ethernet mají 48 bitů dělených do šesti osmibytových čísel v šestnáctkové soustavě. Většinou se zapisuje ve tvaru `AB:CD:EF:01:23:45`, v některých podřadných operačních systémech (Windows) se místo dvojteček píší spojovníky.

Pokud chceme komunikovat na **lokální síti**, použijeme MAC adresy. Jak jsme ale již zjistili, k identifikaci počítačů používáme výhradně IP adresy. Je to z jednoduchého důvodu – IP adresy nám jedním formátem dovolují adresovat jak zařízení na lokální síti tak na síti mimo ni. Jak tedy zjistit MAC adresu zařízení na lokální síti, se kterým si chceme povídat? Použijeme protokol ARP.

ARP

Anglický název protokolu, tedy Address Resolution Protocol (ARP) říká v podstatě vše. Protokol nám umožňuje zeptat se zařízení se známou IP adresou na jeho MAC adresu, abychom mu mohli posílat data. Je důležité říci, že ARP je protokolem vrstvy **síťové** (protože pracuje i s *protokolovými adresami*, v tomto případě IP adresami).

Jak se ale můžeme zařízení zeptat na jeho MAC adresu, když *neznáme* jeho MAC adresu? K tomu slouží speciální *broadcast* MAC adresa `FF:FF:FF:FF:FF:FF`. Pokud se nějaká data pošlou na tuto MAC adresu, rozešlou se **všem** zařízením v síti. Pokud je v síti i zařízení s IP adresou, jehož MAC adresu chceme zjistit, náš broadcast požadavek jistě zachytí a pošle nám odpověď s jeho MAC adresou. Dotazující počítač zasílá v požadavku jak svoji IP adresu tak adresu MAC, takže odpovídající zařízení ví, kam má odpověď poslat.

Síťová vrstva

Tato vrstva je zodpovědná za *směrování*, tedy přenos dat mezi routery (tedy asi to nejdůležitější, co bychom od přenosu dat na síti očekávali) a získávání informací o zařízeních s určitou adresou. Právě na síťové vrstvě se k adresaci používají nám již známé IP adresy.

> **Připomeňme si**
>
> IP adresa jednoznačně identifikuje počítač připojený k nějaké počítačové síti, který využívá IP protokol. Další důležité využití je i to, že nám umožňuje komunikovat s počítači **mimo naši lokální síť**.

Důležitými protokoly na síťové vrstvě jsou IP, ARP a ICMP. O IP a ICMP si povíme právě v této kapitolce.

Sítě dělíme na dva druhy – LAN (Local Area Network) a WAN (Wide Area Network). LAN sítím se říká *lokální síť*, WAN nazývejme její zkratkou. Příklad lokální sítě může být třeba ta vaše domácí – je na ní

pár počítačů, telefonů, tiskáren. To je vše. Všechny tyto zařízení jsou nějakým způsobem připojené k vašemu routeru, který dělí vaši síť od další. Dalším příkladem lokální sítě může být síť nějaké firmy, školy nebo jiného podniku. WAN sítě jsou rozlohou podstatně větší, tou největší je samotný internet. Poskytovatelé internetového přípojení tedy nedělají nic jiného, než že vaši malou síťku za měsíční poplatek připojí pomocí své infrastruktury k sítí větší.

Pojďme se nejdřív podívat na to, jak to vypadá na síťové vrstvě, když se ptáme vzdáleného serveru (náš požadavek tedy bude cestovat přes hranice naší sítě).

Představme si následující situaci:

Vidíme v ní náš počítač, tři routery a server, na který požadavek posíláme. Nejprve musíme překonat cestu z našeho počítače na náš router. Jak ale zjistíme adresu routeru? Při připojení na síť (zastrčení kabelu do portu síťové karty, připojení na WiFi síť) se na našem počítači uloží *default gateway* – IP adresa routeru, který dělí naši síť od té další. Proč se ale tento údaj nejmenuje jednoduše např. router address? Název nese důležitou informaci – pokud se chceme dostat pryč z naší sítě (skrz nějakou virtuální bránu, kterou opouštíme naši malou bezpečnou síťku), můžeme použít IP adresu *default **gateway***.

Dále musíme zjisit MAC adresu routeru, abychom mu mohli poslat náš požadavek (aby ho následně mohl přeposlat dále). To už ale umíme pomocí protokolu ARP. Vyšleme tedy ARP požadvek, router nám na něj odpoví, vezmeme MAC adresu routeru a náš požadavek, který chceme poslat serveru, pošleme routeru.

Router požadavek přijme. Teď ho čeká důležité rozhodnutí. Jak zjistí, jestli požadavek posílat ven ze sítě nebo ne? Potřebujeme nějakým způsobem zjistit, jestli IP adresa příjemce přísluší stejné síti, nebo ne. K tomu slouží další parametr – maska sítě (angl. *subnet mask*). Ta určuje, jaký rozsah IP adres patří do naší sítě. Pro IPv4 adresy vypadá maska stejně, jako IP adresy samotné. V případě první sítě by maska byla `255.255.255.0`. IP adresu i masku sítě můžeme zapsat binárně a podívat se, které bity IP adresy mají na svojí pozici jedničku v masce sítě. Router tedy provede logický AND a dostane číslo, které říká: „Pokud mají dvě IP adresy tuto část stejnou, jsou na stejné síti!“ Pokud se tedy shodují, router vyšle požadavek zpět do naší sítě příjemci. Pokud ne, paket se vydává na nebezpečnou cestu mimo naši síť.

Jsme tedy na druhém routeru. To může být router našeho poskytovatele. Ten se musí rozhodnout, kam požadavek pošle. Je důležité si uvědomit, že tyto routery už nemají jen dvě připojení – mohou být klidně připojené na desítky dalších routerů. Jak se má ale rozhodnout? Má k dispozici **směrovací tabulku** (angl. *routing table*). V ní jsou záznamy o tom, kam se mají vyslat pakety, které routeru přijdou. Tři důležité sloupce směrovací tabulky jsou cíl, maska a brána.

Přidělování IP adres

1.3.4.1 Přílohy