EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Vysoká škola báňská – Technická univerzita Ostrava

INTERNET A SÍTĚ

učební text

Marek Babiuch

Ostrava 2010

Recenze: Ing. Pavel Buřil
Mgr. Jan Veřmiřovský

Název: Internet a sítě
Autor: Marek Babiuch
Vydání: první, 2010
Počet stran: 162

Vydavatel a tisk: Ediční středisko VŠB – TUO

Studijní materiály pro studijní obor Automatické řízení a inženýrská informatika fakulty strojní
Jazyková korektura: nebyla provedena.

Určeno pro projekt:

Operační program Vzděláváním pro konkurenceschopnost
Název: Personalizace výuky prostřednictvím e-learningu
Číslo: CZ.1.07/2.2.00/07.0339
Realizace: VŠB – Technická univerzita Ostrava
Projekt je spolufinancován z prostředků ESF a státního rozpočtu ČR

ISBN 978-80-248-2566-3

OBSAH

POKYNY KE STUDIU ... 4
1. RYCHLÝ ÚVOD DO TVORBY WEBOVÝCH STRÁNEK ... 7
1.1 PRINCIP ZOBRAZENÍ WEBOVÝCH STRÁNEK V PROHLÍŽEČI ... 7
1.2 DOSTUPNÉ PROHLÍŽEČE A NÁSTROJE PRO TVORBU WEB STRÁNEK ... 8
1.3 JAZYKY A STYLY ... 10
1.4 ZDROJOVÝ KÓD STRÁNKY A KOSTRA DOKUMENTU ... 11
1.5 NAŠE PRVNÍ WEBOVÁ STRÁNKA ... 12
2. ELEMENTY JAZYKA HTML ... 15
2.1 ATRIBUTY HTML ELEMENTŮ ... 15
2.2 BARVY V HTML STRÁNKÁCH ... 16
2.3 HLAVIČKA HTML DOKUMENTU ... 17
2.4 FORMÁTOVÁNÍ TEXTU ... 17
2.5 TABULKY, OBRÁZKY, SEZNAMY A ODKAZY ... 19
3. KASKÁDOVÉ STYLY ... 26
3.1 DŮVODY PRO POUŽITÍ KASKÁDOVÝCH STYLŮ ... 26
3.2 ZPŮSOBY POUŽÍVÁNÍ KASKÁDOVÝCH STYLŮ ... 27
3.3 SYNTAXE FORMÁTOVÁNÍ CSS ... 28
3.4 SELEKTORY KASKÁDOVÝCH STYLŮ ... 28
3.5 UKÁZKA POUŽITÍ KASKÁDOVÝCH STYLŮ ... 29
4. LAYOUT WEBOVÝCH STRÁNEK A POZICOVÁNÍ ... 38
4.1 RŮZNÉ FORMY LAYOUTŮ ... 38
4.2 TABULKY, RÁMCE A CSS STYLY ... 39
4.3 CSS POZICOVÁNÍ ... 40
4.4 VYTVOŘENÍ CSS MENU ... 43
4.5 POUŽÍVÁNÍ DOSTUPNÝCH ŠABLON PRO NÁVRH VLASTNÍ APLIKACE ... 45
5. TVORBA WEBOVÝCH STRÁNEK POMOCÍ POKROČILÝCH NÁSTROJŮ ... 50
5.1 POUŽITÍ PROFESIONÁLNÍHO EDITORU ... 50
5.2 POKROČILÉ WEBOVÉ STRÁNKY POMOCÍ CSS ŠABLON ... 52
5.3 VÝUKOVÝ SERVER EXPRESSION WEB 3 STUDIA ... 54
5.4 CSS A XHTML VALIDACE ... 54
6. POUŽITÍ JAVASCRIPTU ... 58
6.1 ÚVOD DO POUŽITÍ JAVASCRIPTU V HTML STRÁNKÁCH ... 58
6.2 PROMĚNNÉ A OPERÁTORY ... 59
6.3 SYNTAXE PŘÍKAZŮ JAVASCRIPTU ... 60
6.4 OBJEKTOVÝ MODEL A VYBRANÉ OBJEKTY ... 61
6.5 PRAKTICKÉ PŘÍKLADY ... 65
7. ZÁKLADNÍ ELEMENTY KOMUNIKACE V POČÍTAČOVÝCH SÍTÍCH ... 71
7.1 ZÁKLADY KOMUNIKACE V POČÍTAČOVÉ SÍTI ... 71
7.2 KOMPONENTY POČÍTAČOVÉ SÍTĚ ... 72

7.3 TYPY ZAŘÍZENÍ ... 73
7.4 KOMUNIKAČNÍ MÉDIA ... 74
7.5 ROZDĚLENÍ SÍTÍ ... 76
7.6 KOMUNIKAČNÍ PROTOKOL ... 77
8. ISO OSI A TCP/IP MODEL ... 81
8.1 VZÁJEMNÁ INTERAKCE PROTOKOLŮ ... 81
8.2 ISO - OSI MODEL ... 82
8.3 TCP/IP MODEL ... 83
8.4 POROVNÁNÍ MODELŮ ISO OSI A TCP/IP ... 84
8.5 KOMUNIKAČNÍ PROCES ... 85
8.6 PROCES ZAPOUZDŘENÍ A ROZBALENÍ PAKETU ... 86
9. APLIKAČNÍ A TRANSPORTNÍ VRSTVA ... 91
9.1 ÚLOHA APLIKAČNÍ VRSTVY ... 91
9.2 FUNKCE APLIKAČNÍCH PROTOKOLŮ ... 92
9.3 APLIKAČNÍ PROTOKOLY A JEJICH SLUŽBY ... 94
9.4 ÚLOHA TRANSPORTNÍ VRSTVY ... 98
9.5 TCP A UDP ... 99
9.6 ADRESOVÁNÍ PORTŮ ... 100
10. IP ADRESOVÁNÍ ... 105
10.1 DEFINICE IP ADRESY ... 105
10.2 PŘEVODY MEZI BINÁRNÍ A DESÍTKOVOU SOUSTAVOU ... 106
10.3 PŮVODNÍ ROZDĚLENÍ IP ADRES ... 107
10.4 MASKA SÍTĚ ... 108
10.5 UNICAST, BROADCAST A MULTICAST ADRESACE ... 110
10.6 SPECIÁLNÍ TYPY IP ADRES ... 111
10.7 ADRESOVÁNÍ PODSÍTÍ ... 112
11. KOMUNIKACE NA ÚROVNI SÍŤOVÉ VRSTVY ... 117
11.1 ÚLOHY SÍŤOVÉ VRSTVY ... 117
11.2 CHARAKTERISTIKA IP PROTOKOLU ... 118
11.3 DEFAULT GATEWAY – BRÁNA DO VENKOVNÍ SÍTĚ ... 120
11.4 SÍŤOVÁ CESTA A SMĚROVACÍ TABULKA ... 121
11.5 STATICKÉ A DYNAMICKÉ SMĚROVÁNÍ ... 122
12. FYZICKÉ PROPOJENÍ, ETHERNET, MAC ADRESY ... 128
12.1 ÚLOHA LINKOVÉ VRSTVY ... 128
12.2 RÁMEC LINKOVÉ VRSTVY ... 129
12.3 ÚLOHA FYZICKÉ VRSTVY ... 130
12.4 ETHERNETOVÁ MAC ADRESA ... 131
12.5 PROPOJOVÁNÍ ZAŘÍZENÍ ... 132
KLÍČ K ŘEŠENÍ ... 141
REJSTŘÍK ... 160

POKYNY KE STUDIU

Internet a sítě

Pro předmět Internet a sítě 1. semestru oboru Automatické řízení a inženýrská informatika ste obdrželi studijní balík obsahující:

- integrované skriptum pro distanční studium obsahující i pokyny ke studiu
- CD-ROM s doplňkovými animacemi vybraných částí kapitol
- harmonogram průběhu semestru a rozvrh prezenční části
- rozdělení studentů do skupin k jednotlivým tutorům a kontakty na tutory
- kontakt na studijní oddělení

Prerekvizity

Pro studium tohoto předmětu se nepředpokládají znalosti z nějakého předmětu zaměřeného na IT technologie. Důležité je, aby student, který chce absolvovat tento předmět, měl základy práce s počítačem, měl jisté zkušenosti s používáním internetových stránek, dokázal nainstalovat nějaký podpůrný program a dokázal najít v operačním systému základní nastavení počítačové sítě či spustit příkazovou řádku. Znalost použití internetového prohlížeče a souvisejících pojmů s ovládáním windows aplikací je samozřejmostí.

Cílem předmětu

je seznámení se s tvorbou webových aplikací pomocí HTML, CSS, javascriptu a naučit studenty základy počítačových sítí. Učební text je rozdělen do dvanácti vybraných kapitol tak, aby na sebe logicky navazovaly a poskytly studentům stavební kámen pro tvorbu webových aplikací a pochopení technologií počítačových sítí. Učební text a celý předmět svým rozsahem nemůže konkurovat mnohasetstránkovým knihám, není to ani jeho cílem, naopak má za pomoci vhodně zvolených řešených příkladů podnítit zájem o tuto mladou avšak robustní a perspektivní oblast.

Pro koho je předmět určen

Modul je zařazen do bakalářského studia oboru Aplikovaná informatika a řízení a je zároveň celofakultním volitelným předmětem.

Předmět Internet a sítě je volitelným předmětem a jeho výuka v denní formě probíhá výhradně formou cvičení. Proto je učební text koncipován tak, aby studentům kombinované formy co nejvíce přiblížil reálný semestr studenta v prezenční formě.

Skriptum se dělí na části a kapitoly, které odpovídají logickému dělení studované látky, ale nejsou stejně obsáhlé. Předpokládaná doba ke studiu kapitoly se může lišit u studentů, kteří již mají bohaté zkušenosti s tvorbou webových stránek a u studentů, kteří jsou pouze začátečníci.

V průběhu semestru bude kladen důraz na osobní přístup pedagoga ke studentům, uvedené kapitoly budou prakticky procvičovány a získané zkušenosti budou využity k tvorbě semestrálního projektu, s jehož zadáním budou studenti seznámeni během semestru.

V každé kapitole je procvičována řada ukázkových příkladů, které budou ještě podrobněji vytvářeny v přiložených animacích. Každá kapitola obsahuje důležité otázky, jejichž odpovědi jsou v klíči k řešení a několik samostatných příkladů vycházejících z probíraných témat.

Při studiu každé kapitoly doporučujeme následující postup:

Čas ke studiu: x hodin

Na úvod kapitoly je uveden **čas** potřebný k prostudování látky. Čas je orientační a může vám sloužit jako hrubé vodítko pro rozvržení studia celého předmětu či kapitoly. Někomu se čas může zdát příliš dlouhý, někomu naopak. Jsou studenti, kteří se s touto problematikou ještě nikdy nesetkali a naopak takoví, kteří již v tomto oboru mají bohaté zkušenosti.

Cíl: Po prostudování tohoto odstavce budete umět

- popsat ...
- definovat ...
- vyřešit ...

Ihned potom jsou uvedeny cíle, kterých máte dosáhnout po prostudování této kapitoly – konkrétní dovednosti, znalosti.

Výklad

Následuje vlastní výklad studované látky, zavedení nových pojmů, jejich vysvětlení, vše doprovázeno obrázky, tabulkami, řešenými příklady, odkazy na animace.

Řešený příklad

Zadání a řešení praktického příkladu jako součást výukového textu.

Korespondenční úkol

Zadání domácí úlohy nebo souvisejícího úkolu v rámci výkladu.

Pojmy k zapamatování

Touto ikonkou je upozorněno na pojem, který bychom si měli pamatovat. Neznamená to ale, že ostatní pojmy výkladu si pamatovat nemusíme.

Další zdroje

Seznam další literatury, www odkazů apod. pro zájemce o dobrovolné rozšíření znalostí popisované problematiky. Budou uvedeny na konci učebního textu.

CD-ROM

Informace o doplňujících animacích, které si může student vyvolat z CD-ROMu připojeného k tomuto materiálu.

Shrnutí kapitoly

Na závěr kapitoly jsou zopakovány hlavní pojmy, které si v ní máte osvojit. Pokud některému z nich ještě nerozumíte, vraťte se k nim ještě jednou.

Kontrolní otázka

Na konci každé kapitoly bude zveřejněn seznam otázek, které se mohou v plném nebo modifikovaném znění objevit u zkoušky z předmětu Programování aplikací pro Internet II. Tyto otázky jsou také podkladem ke státnicovým otázkám magisterského oboru Automatické řízení a inženýrská informatika, konkrétně státnicového předmětu Informační technologie.

Úkol k řešení

Na konci kapitol, které kromě teoretických otázek umožňují i praktické procvičení probraného učiva, najdete úkol k řešení. Rovněž s tímto úkolem se můžete v obdobném znění setkat u zkoušky či zápočtového testu.

Klíč k řešení

Odpovědi z kontrolních otázek výše jsou uvedeny v závěru učebnice v Klíči k řešení. Používejte je až po vlastním vyřešení úloh, jen tak si samokontrolou ověříte, že jste obsah kapitoly skutečně úplně zvládli.

1. RYCHLÝ ÚVOD DO TVORBY WEBOVÝCH STRÁNEK

Čas ke studiu: 2,5 hodiny

Cíl Po prostudování tohoto odstavce budete umět

- vysvětlit princip načtení stránky do prohlížeče
- popsat současný stav ve světě internetových prohlížečů
- najít vhodný editor html stránek
- zobrazit zdrojový kód stránky v editoru
- vytvořit kostru html stránky
- popsat základní vlastnosti html jazyka
- vytvořit svou první html stránku

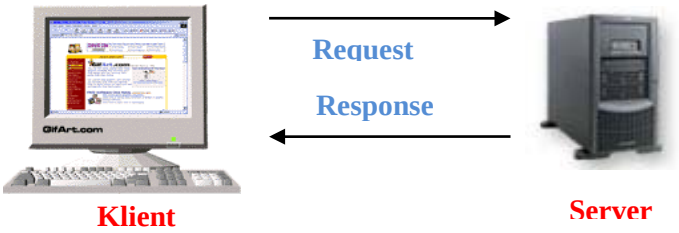
Výklad

Tento učební text není manuálem jazyka ani programátorskou příručkou, dal by se spíše charakterizovat jako interaktivní průvodce tvorby webových stránek a základů práce v počítačové síti. Přesto, že se nejedná o rozsáhlou programátorskou příručku, můžeme říci, že po nastudování tohoto učebního modulu budeme schopni vytvářet robustní a kvalitní webová sídla a výborně se orientovat v prostředí počítačových sítí. V úvodní kapitole probereme stručně princip zobrazení webových stránek v prohlížeči a přesuneme se na dostupné nástroje pro jejich tvorbu.

1.1 Princip zobrazení webových stránek v prohlížeči

Webové stránky jsou pro mnoho uživatelů podstatou internetu. V raných dobách použití internetové sítě existovaly stránky pouze ve statické podobě. Uživatelé neměli možnost pomocí webových stránek zpracovávat a měnit žádné informace. Postupně se pomocí skriptovacích jazyků a xhtml měnily stránky na dynamické, přibylo mnoho nových prvků a formulářů. Po zavedení serverového zpracování přibyly další široké možnosti uplatnění dynamických stránek včetně spolupráce s databázemi. Serverové zpracování skriptů však nebude náplní tohoto učebního textu, budeme se věnovat tvorbě moderních statických i dynamických stránek se zpracováním na straně klienta.

Komunikace počítače s internetovým prohlížečem na straně jedné a webovým serverem poskytujícím výslednou webovou stránku na straně druhé probíhá v modelu klient/server. Tento model komunikace pracuje způsobem žádost/odpověď (*request/response*). V praxi to znamená, že klientský počítač zašle serverovému požadavek na informace. Serverový počítač klientovi odpoví a informace mu poskytne. Klientský počítač pak získané informace zobrazí v prohlížeči.



Obr 1.1 Komunikace typu klient/server

Vše je tedy založeno na principu žádost/odpověď, neexistuje žádné trvalé spojení, jedná se o dva samostatně běžící procesy na počítačích. Server nemá žádné tušení o tom, co se právě na klienském počítači děje.

Pojem k zapamatování: Model klient/server

Klient/server model je základní myšlenkou síťové technologie. Používá se v síťových protokolech jako http, smtp, dhcp, dns a další. Nejpoužívanějším klientem je a vždy bude webový prohlížeč. Každý klient může posílat žádosti jednomu nebo více serverům, naopak každý server vrací odpověď na konkrétní požadavek klienta.

1.2 Dostupné prohlížeče a nástroje pro tvorbu web stránek

Webové stránky se zobrazují v prohlížečích tzv. *browserech*, které umožňují *http* komunikaci s webovým serverem a přijatý kód zobrazují ve formě webových stránek. V současné době existují desítky prohlížečů, které jsou založeny na různých jádrech. Jádro prohlížeče slouží ke grafickému vykreslování obsahu stránek a k tomuto účelu samotné jádro mohou používat i jiné aplikace než jen prohlížeče. V současné době se prohlížeče stavějí na třech nejznámějších konkurenčních jádrech a to jsou jádra Trident, Gecko a WebKit. Jader je samozřejmě také více, na své javascriptové jádro Caracan je v současné době hrdá norská firma Opera Software, která svůj prohlížeč deklaruje v mnoha nezávislých testech jako nejrychlejší.
Na prvně zmiňovaném jádře Trident je založen Microsoft Internet Explorer, na jádru Gecko konkurenční a stále více používaná Mozilla Firefox. Třetí jádro WebKit je základem Applového prohlížeče Safari, který ale také běží i na platformách Windows a hlavně jednoho z nejmladších prohlížečů Chrome od společnosti Google.

Obr. 1.2 Ikony nejznámějších prohlížečů

Jádra prohlížečů určují rozdílné vykreslovací schopnosti prohlížečů, některé prvky jsou odlišné i v bezpečnostním přístupu. Nejen těmito vlastnostmi se uvedené prohlížeče liší. V dnešní a budoucí době bude o popularitě prohlížečů rozhodovat nejen bezpečnost, ale také jednotlivé vložené nové moduly, tzv. *pluginy*, které uživatelům nabízejí určitý komfort v oblasti, kterou využívají. Některé prohlížeče, tak přímo komunikují prostřednictvím webových služeb s různými servery, jako například Flickr pro sdílení a správu fotografií, jiné prohlížeče obsahují podporů různých komunikačních programů a messengerů apod. Proto nelze jednoznačně určitý prohlížeč preferovat před jiným a proto také vývojáři webových aplikací musí dbát na to, aby jejich aplikace pod těmito prohlížeči fungovaly. Nesmíme opominout ani fakt, že některé prohlížeče jsou přímo výhodné pro použití v jiných typech operačních systému rozdílných od platformy Windows a také na rostoucí trend mobilních zařízení s možností připojení k internetu.

Pojem k zapamatování: http protokol

Hypertext transfer protokol je nejpoužívanějším síťovým protokolem, který zprostředkovává komunikaci mezi webovým serverem a klientskou aplikací, kterou reprezentuje webový prohlížeč. Činnost protokolu je charakterizována jako série zpráv typu dotaz – odpověď. Klientský počítač se dotazuje a webový server na tyto dotazy odpovídá.

Právě pro výše uvedené skutečnosti již Internet Explorer není tak dominantním prohlížečem, který měl v minulosti podíl okolo 95%. V roce 2010 je to již jen více než 60%, přesto má před druhým

nejvýznamnějším prohlížečem Mozilla Firefox s více než 20% stále velký náskok. Další významné prohlížeče, které máme zobrazeny na obrázku: Opera, Google Chrome a Safari dosahují každý okolo 5%, ale tyto údaje se mohou každým měsícem měnit.

V prvním odstavci této kapitoly jsme si definovali, v jakém programu můžeme stránky prohlížet, nyní nastal čas zodpovědět otázku, v jakém programu webové stránky vytvářet? Odpovědí na tuto otázku můžeme opět slyšet desítky. Rozhodujícím aspektem je právě naše pozice v roli tvůrce stránek. Zdrojový kód stránek můžeme totiž psát i v obyčejném textovém editoru jako je *notepad*. Nebude to příliš pohodlné, ale jsou případy, kdy si s tímto editorem pohodlně vystačíme. Pro relevantní odpověď na otázku musíme vědět, zdali se tvorbou stránek hodláme věnovat dlouhodobě, profesionálně či amatérsky a jaké webové stránky budeme vytvářet. Mnohé editory jsou volně šiřitelné zdarma, naopak ty licencované nejsou levnou záležitostí, avšak ve svém instalačním balíku dokáží nabídnout profesionální funkce. Mezi volně šiřitelné editory patří velice oblíbený editor PSPad. Na opačném konci stojí například profesionální nástroj Dreamweaver společnosti Macromedia – dnes již Adobe. V našem předmětu se seznámíme i s Microsoftími produkty Visual Studio a převážně Microsoft Expression Web, který je nástupcem bývalého nástroje Front Page z balíku Office této společnosti.

Obr. 1.3 Výkonný nástroj Expression Web 3 Studio pro tvorbu webových stránek

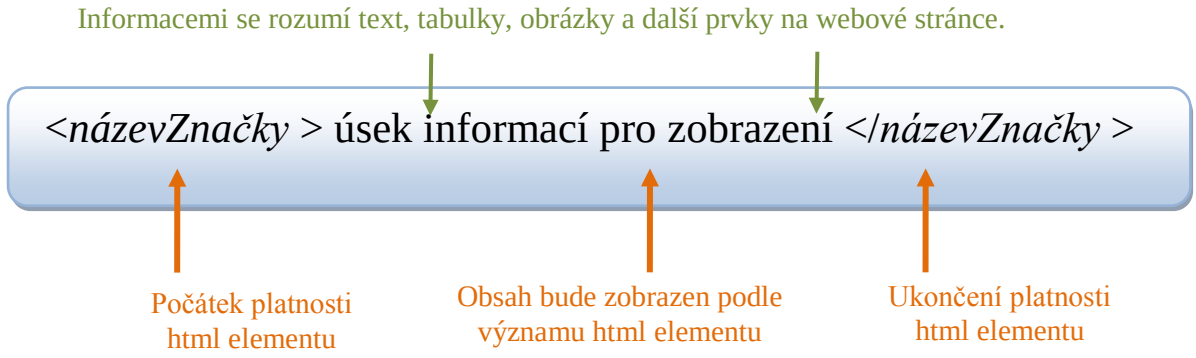
Seznam některých volně dostupných dostupných editorů HTML

1. PSPad Editor - http://www.pspad.com/cz/
2. HTML Kit - http://www.chami.com/html-kit/
3. HTML Editor - http://editor.2b.cz/
4. Dynamic HTML Editor - http://www.dynamic-html-editor.com/en/home.asp
5. Golden HTML Editor - http://www.golden-html.com/
6. CofeeCup Html Editor 2008 - http://www.coffeecup.com/
7. 1Site Lite - http://www.visualvision.com/software/1site_e.html?1site
8. Hypertext Builder - http://www.paksoft-productions.com/hb/hb1.asp
9. AceHtml - http://software.visicommedia.com/en/products/acehtmlfreeware/
10. Webster - http://www.dwn.cz/webster

A mnoho dalších, záměrně neuvádím čísla verzí programů, neboť editory se neustále zdokonalují. Jejich web adresy výrobců by však měly zůstat stále stejné.

1.3 Jazyky a styly

Pro vývoj webových stránek se používají značkovací jazyky, skriptovací jazyky a kaskádové styly. Značkovací jazyk HTML (H*yperText Markup Language*) vychází ze standardu SGML (*Standard Generalized Markup Language*). SGML je univerzální značkovací jazyk dán ISO standardem a umožňuje definovat různé typy značkovacích jazyků podle své syntaxe a pravidel. Značkovací jazyk je to proto, že na rozdíl od programovacích jazyků, kdy určité konstrukci klíčových slov rozumí kompilátor programu, zde pomocí značek se oznámí cíli, jak má daný úsek informací mezi značkami zobrazit. HTML je tedy jazyk, který definuje své značky, které mají význam pro internetový prohlížeč. Značky jsou definovány jako klíčová slova či zkratky z anglického jazyka, uzavřené mezi ostré závorky.



Obr. 1.4 Syntaxe značkovacího jazyka

V praxi to znamená, že prohlížeč zobrazí informace mezi počáteční a koncovou značkou podle významu dané značky. Například značka `<b>` formátuje písmo tučně a proto vše mezi touto a koncovou značkou `</b>` zobrazí prohlížeč tučným písmem. Následující kapitola se věnuje významům nejpoužívanějších html značek.

Dalšími jazyky, které pomáhají vyvíjet webové stránky, jsou skriptovací jazyky. Dělí se na klientské a serverové. Syntaxe a použití serverových jazyků je mimo rámec učebního textu a můžeme se s nimi setkat v navazujících kurzech Programování aplikací pro Internet I a II. Klientským skriptovacím jazykem je například javascript, který si blíže představíme v kapitole 6.

Mezi nezbytné nástroje pro vývoj webových stránek patří kaskádové styly. Dnešní moderní stránky se bez nich již neobejdou. Jejich posláním je oddělit vzhled od obsahu, což je právě nevýhodou značkovacích jazyků, kdy míchámе informační obsah s určením, jak se má interpretovat. Kaskádové styly naopak určují pouze vzhled a to na pokročilé úrovni. Kaskádové styly dokáží určit html elementům formát a zpřesnit jejich vlastnosti a oddělují tak tyto definice od vlastního informačního obsahu. Kaskádové styly budou obsahem třetí kapitoly učebního textu.

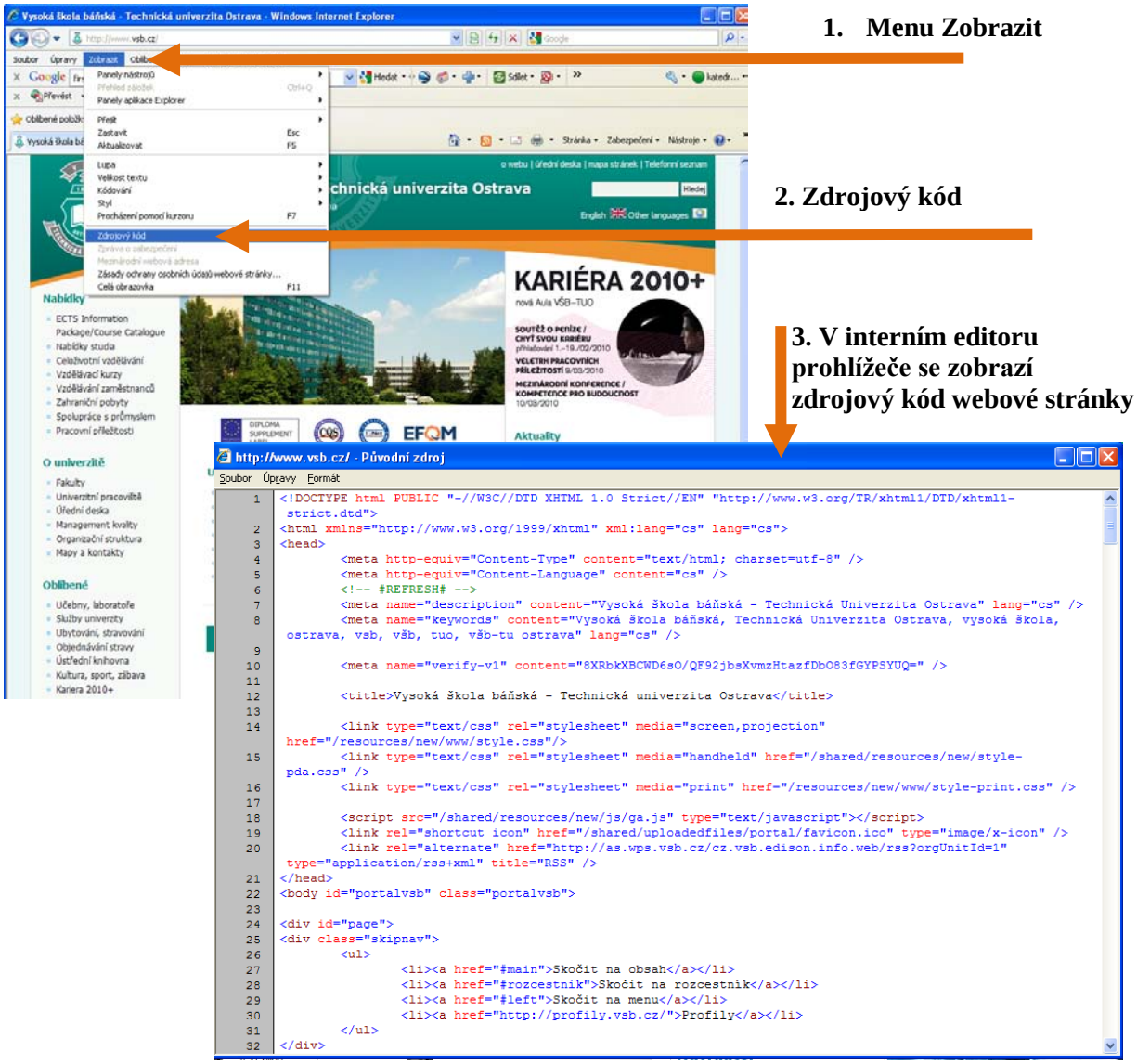
Pojem k zapamatování: HTML jazyk

HTML jazyk je značkovací jazyk pro www. Byl vyvíjen od roku 1989. V roce 1994 byla uveřejněna verze 2.0, která již odpovídá standardům SGML. Verze jazyka 3.0 nebyla přijata jako standard pro svou velkou složitost, až v roce 1996 po opravách byla přijatá verze 3.2

V současné době se používá verze 4.01, která byla vydána již v roce 1999. Na specifikaci verze html 5 se neustále pracuje a pravděpodobně bude uvolněna v roce 2012 a standardizována deset let poté v roce 2022. Deset let si nechávají vývojáři na odladění všech chyb ve specifikaci.

1.4 Zdrojový kód stránky a kostra dokumentu

V našich cvičeních budeme pracovat tak, že budeme vytvářet a editovat zdrojový kód html stránky, který budeme ihned prohlížet ve webovém prohlížeči. Zdrojový kód nám umožní prohlédnout každý typ prohlížeče. Musíme zvolit z nabídkové lišty prohlížeče menu → Zobrazit → Zdrojový kód. Postupně si probereme jednotlivé html elementy (viz. následující kapitola) a budeme je umísťovat do zdrojového kódu, v editoru výsledný kód uložíme a v prohlížeči obnovíme (*refresh*) obsah stránky.



Obr. 1.5 Zobrazení zdrojového kódu webové stránky

Bývá zvykem v každé vývojářské literatuře při popisu základních vlastností, než ještě čtenář pronikne do popisů instrukcí a příkazů, ukázat první jednoduchý příklad. V našem případě to bude zobrazení jedné webové stránky, která bude mít svůj název, bude obsahovat dva řádky textu a jeden obrázek. Abychom tuto první triviální stránku vytvořili, potřebujeme znát kostru html dokumentu. Kostra dokumentu je obecný tvar nejnutnějších elementů, které budou ve zdrojovém kódu vždy, i kdyby se jednalo o prázdnou stránku. Kostra dokumentu obsahuje párové elementy `<html> </html>`, `<head></head>` a `<body> </body>`. Html dokument obsahuje hlavičku a tělo. V těle dokumentu je hlavní informační obsah, v hlavičce dokumentu je její název zobrazen mezi elementy `<title>` a `</title>`. Před samotným začátkem dokumentu je ještě definice `<!DOCTYPE >`, která podle specifikací W3C konsorcia udává typ dokumentu a je důležitá při různých validacích a automatických činnostech, které v tuto chvíli vůbec nemusíme znát a používat.

1.5 Naše první webová stránka

K napsání kódu stránek, které budeme v prvních kapitolách vysvětlovat, nám postačí i editor Notepad. Na obrázku je však použit pokročilejší editor proto, aby nám jednotlivé elementy zpřehlednil a zpříjemnil barevným odlišením. Než vytvoříme slibovanou stránku s obrázkem, musíme ještě obrázek umístit do stejného adresáře, kde založíme zdrojový kód stránky, jinak bychom k němu museli psát cestu.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Kostra HTML stránky</title>
</head>
<body>
Ahoj tohle je moje první HTML stránka <br />
Na další řádek umístím obrázek <br />
<img src="obrazek.jpg" alt="foto" />

</body>
</html>
```

Obr. 1.6 Zdrojový kód naší první webové stránky

Zdrojový kód můžeme, jak už bylo řečeno napsat bez zmíněného elementu `<!DOCTYPE >`. Nyní můžeme stránku uložit pod libovolným názvem s příponou htm či html a dvojklikem ji zobrazíme v prohlížeči.

Shrnutí kapitoly

Komunikace počítače s internetovým prohlížečem na straně jedné a webovým serverem poskytujícím výslednou webovou stránku na straně druhé probíhá v modelu klient/server. Tento model komunikace pracuje způsobem žádost/odpověď (*request/response*). V praxi to znamená, že klientský počítač zašle serverovému požadavek na informace. Serverový počítač klientovi odpoví a informace mu poskytne. Klientský počítač pak získané informace zobrazí v prohlížeči.

V současné době existují desítky prohlížečů, které jsou založeny na různých jádrech. Jádro prohlížeče slouží ke grafickému vykreslování obsahu stránek a k tomuto účelu samotné jádro mohou používat i jiné aplikace než jen prohlížeče. V současné době se prohlížeče stavějí na třech nejznámějších konkurenčních jádrech a to jsou jádra Trident, Gecko a WebKit. Jader je samozřejmě také více, na své javascriptové jádro Caracan je v současné době hrdá norská firma Opera Software, která svůj prohlížeč deklaruje v mnoha nezávislých testech jako nejrychlejší.

Na prvně zmiňovaném jádře Trident je založen Microsoft Internet Explorer, na jádru Gecko konkurenční a stále více používaná Mozilla Firefox. Třetí jádro WebKit je základem Applového prohlížeče Safari, který ale také běží i na platformách Windows a hlavně jednoho z nejmladších prohlížečů Chrome od společnosti Google.

Hypertext transfer protokol je nejpoužívanějším síťovým protokolem, který zprostředkovává komunikaci mezi webovým serverem a klientskou aplikací, kterou reprezentuje webový prohlížeč. Činnost protokolu je charakterizována jako série zpráv typu žádost – odpověď.

Pro vývoj webových stránek se používají značkovací jazyky, skriptovací jazyky a kaskádové styly. Značkovací jazyk HTML (*HyperText Markup Language*) vychází ze

standardu SGML (*Standard Generalized Markup Language*). SGML je univerzální značkovací jazyk dán ISO standardem a umožňuje definovat různé typy značkovacích jazyků podle své syntaxe a pravidel.

Značky jsou definovány jako klíčová slova či zkratky z anglického jazyka, uzavřené mezi ostré závorky: `<názevZnačky>` úsek informací pro zobrazení `</názevZnačky>`.

Kostra dokumentu obsahuje párové elementy `<html> </html>`, `<head></head>` a `<body> </body>`. Html dokument obsahuje hlavičku a tělo. V těle dokumentu je hlavní informační obsah, v hlavičce dokumentu je její název zobrazen mezi elementy <title> a </title>. Před samotným začátkem dokumentu je ještě definice `<!DOCTYPE >`, která podle specifikací W3C konsorcia udává typ dokumentu.

Úkol k řešení 1.1 – Instalace druhého prohlížeče web stránek

Pro tvorbu webových stránek a jejich testování je vždy výhodné mít nainstalované minimálně dva prohlížeče. Pokud používáte pouze jeden, neváhejte nainstalovat jiný, doporučuji nějaký z trojice Firefox, Opera, Google Chrome. Instalace proběhne bez potíží a je zcela zdarma. Po instalaci si vyzkoušejte základní práci s novým prohlížečem.

Úkol k řešení 1.2 – Práce v prohlížečích a zdrojový kód stránky

Práce v různých prohlížečích je v základních funkcích totožná, ale přesto mohou mít prohlížeče odlišné nástroje. Projděte si v různých prohlížečích možnosti nastavení, zobrazení nových záložek, nastavení domovské stránky a hlavně zobrazení zdrojového kódu právě prohlížené stránky.

Úkol k řešení 1.3 – První stránka

Vytvořte svou první web stránku, tím že vytvoříte v libovolném editoru kostru html dokumentu a přidáte libovolný text či obrázek jako v příkladu kapitoly 1.5.

Kontrolní otázka 1.1

Jaký je rozdíl mezi jednotlivými prohlížeči? Proč je jich takové množství?

Kontrolní otázka 1.2

Které prohlížeče, jež nebyly v úvodní kapitole vyjmenovány, dále existují?

Kontrolní otázka 1.3

Co je to SGML?

Kontrolní otázka 1.4

Jak se vyvíjel jazyk html?

Kontrolní otázka 1.5

Které elementy jsou nepárové? Jak se tedy řeší jejich ukončení?

Kontrolní otázka 1.6

Které elementy html bude obsahovat webová stránka vždy?

Kontrolní otázka 1.7

Jakým způsobem komunikace se zobrazují stránky v prohlížeči?

Kontrolní otázka 1.8

Jaké nástroje používáme k tvorbě html kódu?

2. ELEMENTY JAZYKA HTML

Čas ke studiu: 3 hodiny

Cíl Po prostudování této kapitoly budete umět

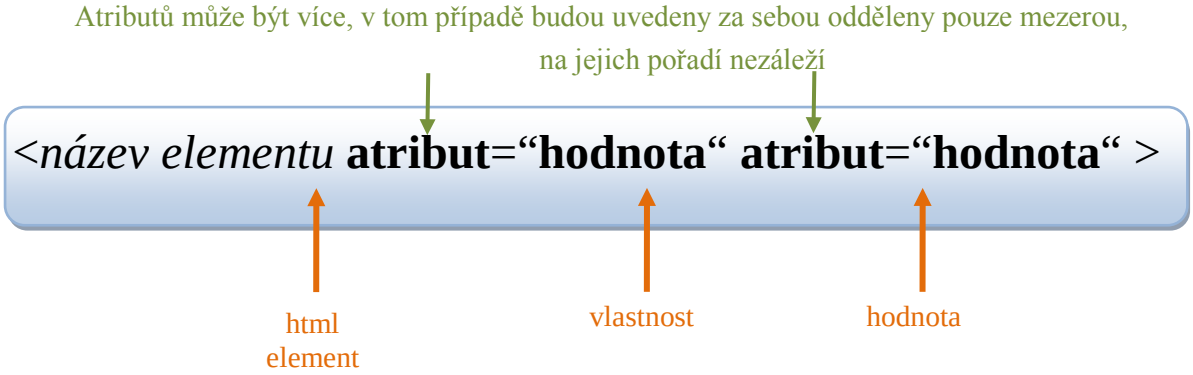
- syntaxi jazyka HTML
- vyhledat a použít všechny elementy jazyka
- definovat volitelné elementy v hlavičce webové stránky
- přizpůsobovat a formátovat html elementy podle atributů
- formátovat text, seznamy, tabulky, obrázky
- používat odkazy v html stránkách
- definovat a používat barvy v html dokumentech

Výklad

Po úvodní kapitole se rychle vrhneme na syntaxi html elementů. Cílem kapitoly je seznámit se s nejpoužívanějšími elementy html a jejich použitím pochopit princip vývoje web stránek pomocí jazyka html. Po nastudování této kapitoly již pro studenta nebude problém vyhledat a použít veškeré html emementy, tedy i ty, na které se v této kapitole nedostane.

2.1 Atributy HTML elementů

HTML elementy mohou ve své definici používat buďto pouze název elementu v ostrých závorkách, tak jak jsme definovali v minulé kapitole, nebo můžeme použít atributy elementů, které danému elementu určí ještě dodatečné vlastnosti. Těmito vlastnostmi bývá zarovnávání, velikost, barva a mnoho dalších. Syntaxi použití atributů html elementů vidíme na následujícím obrázku a samotné atributy elementů, které upřesňují vlastnosti elementů, budeme probírat celou kapitolu.



Obr. 2.1 Syntaxe použití atributů html elementu

2.2 Barvy v HTML stránkách

V html dokumentech můžeme používat barvy pro běžné písmo, nadpisy, pozadí stránky, odstíny buněk tabulky a další prvky. Barvy jsou vyjádřeny ve formátu RGB (*RedGreenBlue*) pomocí hexadecimálního kódu nebo přímo čísel odstínu od 0 do 255. Při použití hexadecimálního kódu použijeme před číselným vyjádřením barvy znak **#**.

Příklad – definice barevného odstínu

```
<body bgcolor="#C0C0C0" text="rgb(0,0,255)">
```

Přes množství barev, které nám RGB zobrazení nabízí, musíme být obezřetní a použití barev uvážit. Webová stránka musí být barevně vyvážená, nejsou vhodné ostré a kontrastní barvy. Stránka je především určena uživatelům, kteří v ní hledají užitečné informace, a proto by je barvy neměly rušit či dokonce odradit.

Pojem k zapamatování: Počet barev

Reprezentace barev v RGB formátu vyjádřené v hexadecimálním kódu, kdy máme 2^{16} možností pro odstíny červené, zelené a modré vyjadřuje 256 x 256 x 256 možných barev. Můžeme tedy vytvořit 16 777 216 možných barevných odstínů.

Pro použití vhodných barev je vhodné místo experimentu navštívit webové stránky profesionálů, kteří se touto tematikou zabývají, stačí správně formulovat dotaz ve vyhledávači. Přesto si dovolím doporučit jeden online nástroj pro volbu barev, naleznete jej na: http://www.colorsontheweb.com/.

Korespondenční úkol – palety barev

Vyhledejte palety barev pro použití v HTML dokumentech. Například na serveru w3schools.com

Doporučuji také vyhledat stránky s možností návrhu barev s přímým generováním hexakódu barvy, např.: http://colorschemedesigner.com/

Pro základní barvy nemusíme používat číselný *rgb* nebo hexadecimální kód. Základní barvy mají svá jména, samozřejmě v anglickém jazyce. Jména těchto barev můžeme používat přímo v atributech html elementů. Na následujícím obrázku vidíme pro základních 16 barev jejich název, odstín i kódové označení.

Barva	Hexakód	Barva	Hexakód	Barva	Hexakód	Barva	Hexakód
aqua / cyan	#00FFFF	gray	#808080	navy	#000080	silver	#C0C0C0
black	#000000	green	#008000	olive	#808000	teal	#008080
blue	#0000FF	lime	#00FF00	purple	#800080	white	#FFFFFF
fuchsia / magenta	#FF00FF	maroon	#800000	red	#FF0000	yellow	#FFFF00

Obr. 2.2 16 barev, které můžeme použít jmenným označením v HTML dokumentech

2.3 Hlavička HTML dokumentu

Hlavička HTML dokumentu je nezbytnou částí každé stránky. Je uvozena elementy `<head></head>`. Mezi těmito elementy se může nacházet pouze několik elementů, přesto jsou některé z nich nepostradatelnými a budeme je pravidelně používat. Elementy nacházející se v hlavičce jsou: `<title>`, `<link>`, `<base>`, `<meta>`, `<style>`, `<script>` a jejich párové ukončení.

Nejznámějším elementem hlavičky je element *title* definující název stránky. Nepodceňujme tento element, je důležitý pro vyhledávače, které tyto stránky třídí, ale hlavně pro samotného čtenáře webových stránek. S elementem *link* se setkáme při tvorbě externích souborů s kaskádovými styly. Tímto elementem soubor s kaskádovým stylem přilinkujeme do našeho dokumentu. Neuvádím zde ukázku, protože se CSS stylům budeme věnovat celou příští kapitolu, kde bude probrán rovněž element *style*. Důležitým elementem hlavičky je tag *meta*, který popisuje pomocí svých atributů obsah stránky, klíčové slovo pro vyhledávání autora stránek a znakovou sadu, ve které se bude zobrazovat informační obsah webové stránky. Element skript budeme rovněž probírat v jiné kapitole. Na obrázku 2.3 vidíme zdrojový kód hlavičky HTML stránky s příslušnými elementy.

```
<head>
    <title>Učební text předmětu Internet a sítě</title>
    <meta name="description" content="HTML učební text, CSS, XML, a Javascript" />
    <meta name="keywords" content="Internet, sítě, HTML, CSS, XML, JavaScript, TCP, IP" />
    <meta name="author" content="Marek Babiuch" />
    <meta http-equiv="Content-Type" content="text/html;charset=ISO-8859-2" />
    <style type="text/css">

    </style>
    <script type="text/javascript">
    //zde bude javascript
    </script>
</head>
```

Obr. 2.3 HTML hlavička s elementy `<title>`,`<meta>`,`<style>`,`<script>`

2.4 Formátování textu

HTML jazyk se vyvíjí zhruba 20 let, přičemž jeho poslední verze HTML 4.01 se používá již jedno desetiletí. Od prvních verzí se jazyk měnil a mnoho základních elementů se již nepoužívá, přestože jej prohlížeče dokáží interpretovat. S vývojem progresivních CSS stylů, které mnoho elementů a atributů nahradí ve funkčnosti, pohodlí a efektivitě, jsou některé elementy HTML již minulostí. Přesto však jsou elementy, bez kterých si formátování webových prvků nedokážeme představit. Tato kapitola přehledně v tabulce uvede seznam používaných elementů a upozorní na ty, které by již měli být nahrazeny modernějšími metodami použití.

Tab. 2.1 Elementy fyzického formátování

Element	Význam	Použití v současné době
b	tučné písmo	ano
i	kurzíva	ano
u	podtržené písmo	nedoporučuje se
sub	dolní index	ano
sup	horní index	ano
small	zmenšení písma	ano
big	zvětšení písma	ano
s, strike	přeškrknutý text	nedoporučuje se
del	smazaný text	ano, nový element
ins	vložený text	ano, nový element
blink	blikající text	nedoporučuje se
nobr	nezalomený text	nedoporučuje se

Elementy se nedoporučují ze tří důvodů. Prvním důvodem je zastaralost uvedeného elementu. Takový element nemusí být podporován prohlížeči a může dělat při zobrazení problémy. Takovým elementem je například `<font>`, který jsem raději do přehledu formátovacích tagů ani nezařadil. Druhým důvodem pro nedoporučení používat elementy je pozbytí jejich významu při vývoji hypertextových dokumentů. Element `<u>` prohlížeč zobrazí podtrženě, avšak v hypertextových dokumentech bývají podtržené odkazy, ostatní podtržený text je matoucí. Třetím důvodem je nahrazení novějším elementem či dokonce CSS stylem. Elementy `<s>` a `<strike>` nahrazujeme novým elementem `<del>` či CSS stylem *text-decoration:line-through*. Obdobně je tomu u blikajícího textu, ten nahrazujeme CSS stylem *text-decoration:blink*. Zakázané zalomení řádku `<nobr>` nahrazujeme stylem *white-space:nowrap*.

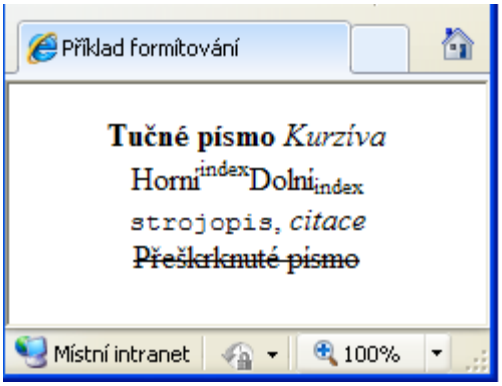
Tab. 2.2 Elementy logického formátování

Element	Význam	Poznámka
span	úsek textu	oddíl textu, který používá obecné atributy
strong	tučné zvýraznění	obdobně jako b, ale logicky znamená zvýrazněný
em	zvýraznění kurzívou	emphasis
cite	citace –kurzíva	citace, která se nezalamuje
code	výpis kódu - strojopis	neproporcionální písmo
dfn	definice termínu –kurzíva	některé prohlížeče zobrazují kurzívu tučně
kbd	vstup z klávesnice - strojopis	málo používaný element
samp	ukázka –strojopis	málo používaný element
var	proměnná – kurzíva	málo používaný element
abbr	ustálený výraz	vysvětlení zkratky, zobrazuje se normálně
acronym	zkratka	zobrazuje se normálně
q	citace	zdroj citace by měl obsahovat atribut cite
tt	teletype text – (dálnopis)	něco jako strojopis – courier font

Logické formátování původně vymezovalo text pro jeho přisouzení významu, nikoliv vzhledu. To znamená, že logicky členilo text bez fyzického formátování. Text se tak mohl členit na obsah, zkratky, citace, proměnné, termíny, výpisy kódu apod. S postupem času se formátování textu na webových stránkách vyvinulo tak, že i logické formátování sebou nese význam formátování fyzického.

```
<p align="center">
<b> Tučné písmo</b>  <i> Kurzíva </i> <br />
Horní<sup>index</sup>Dolní<sub>index</sub><br />
<tt>strojopis</tt>,<cite> citace </cite> <br />
<del> Přeškrknuté písmo </del>
</p>
```

Obr. 2.4 Formátování pomocí elementů html



Obr. 2.5 Zobrazení stránky se zdrojovým kódem z předchozího obrázku

2.5 Tabulky, obrázky, seznamy a odkazy

Samotným textem bychom poutavé stránky nevytvořily. Důležitou součástí stránek jsou obrázky, animace, odkazy, tabulky a další prvky. V této kapitole si vysvětlíme princip formátování tabulek, seznamů, obrázků a odkazů.

Tabulky

Tabulky jsou velmi důležitou součástí webového obsahu. Jsou obsaženy prakticky ve všech portálech a e-shopech, některé z nich jsou viditelné, jiné zase plní roli rozmístění dat na stránce a o jejich existenci na první pohled nevíme. Proto tabulky obsahují celou řadu různých atributů zobrazování, které vidíme přehledně v tabulkách této kapitoly.

Tab. 2.3 Elementy pro zobrazování tabulek

Element	Význam
table	tabulka
tr	řádek tabulky
td	buňka tabulky
th	hlavičková buňka tabulky
caption	popisek tabulky
col, colgroup	skupina sloupců tabulky
tbody	tělo tabulky
thead	hlavička tabulky
tfoot	patička tabulky

Tab. 2.4 Atributy elementu `<table>`

Atribut	Význam	Hodnoty
align	obtékání tabulky textem	left, right, center
cellpadding	vnitřní okraj buněk	v pixelech
cellspacing	vnější okraj buněk	v pixelech
border	šířka rámečku buněk	v pixelech
width	minimální šířka tabulky	šířka v pixelech nebo procentech
height	minimální výška tabulky	výška v pixelech nebo procentech
background	obrázek na pozadí	url obrázku
bgcolor	barva pozadí	barva
bordercolor	barva rámečku	barva
bordercolorlight	světlejší barva rámečku	barva
bordercolordark	tmavší barva rámečku	barva
frame	vykreslení rámečku okolo	void, border, box, hsides, vsides, above, below, khs, rhs
rules	vykreslení mřížky	none, all, rows, cols, groups
summary	stručné shrnutí obsahu	text pro čtečky nevidomých

Tab. 2.5 Atributy elementu `<td>`

Atribut	Význam	Hodnoty
align	horizontální zarovnání buňky	left, center, right, justify
valign	vertikální zarovnání buňky	top, middle, bottom, baseline
width	doporučená šířka buňky	šířka v pixelech nebo procentech
height	minimální výška celého řádku	výška v pixelech
nowrap	obsah buňky nezalamovat	bez hodnoty

background	obrázek pozadí	url obrázku
bgcolor	barva pozadí	barva
bordercolor	barva rámečku	barva
bordercolorlight	světlejší barva rámečku	barva
bordercolordark	tmavší barva rámečku	barva
rowspan	spojení buněk na n řádků	počet řádků
colspan	spojení buněk na n sloupců	počet sloupců

```
<table border="1" style="width: 50%; height: 141px;">
    <tr bgcolor="Silver">
        <td align="center"  valign="middle">
            Zaměstnanec</td>
        <td align="center">
            Osobní číslo</td>
        <td align="center">
            Telefon</td>
    </tr>
    <tr align="center">
        <td>
            František Kusý</td>
        <td>
            kus007</td>
        <td>
            +420 732 111 987</td>
    </tr>
    <tr align="center">
        <td>
            Josef Novák</td>
        <td>
            nov032</td>
        <td>
            +420 732 112 654</td>
    </tr>
</table>
```

Obr. 2.6 Příklad zdrojového kódu pro definici tabulky

Tab. 2.6 Výsledná tabulka z předchozího příkladu

Zaměstnanec	Osobní číslo	Telefon
František Kusý	kus007	+420 732 111 987
Josef Novák	nov032	+420 732 112 654

Obrázky

Obrázky jsou nezbytnou součástí každé webové stránky. Pro vložení obrázku používáme nepárový element `<img>`. Tento element nelze použít bez atributů, neboť povinným atributem je cesta k obrázku. Pokud bude obrázek přímo ve stejném adresáři jako zdrojová html stránka, cesta bude obsahovat pouze název obrázku, jako je tomu na obr. 2.7. Protože obrázků obsahuje většina webových

sídel mnoho, doporučuje se na webovém serveru vytvořit adresář např. *images* a na něj se odkazovat v atributu *src* elementu <img>.

Tab. 2.7 Atributy elementu <img>

Atribut	Význam	Hodtoty
src	umístění souboru s obrázkem	URL obrázku
alt	alternativní popis	text
lowscr	nahrádní obrázek pro malý display	URL obrázku
width	šířka obrázku	šířka v pixelech nebo procentech
height	výška obrázku	výška v pixelech nebo procentech
border	tloušťka rámečku	v pixelech
vspace	vertikální okraj	v pixelech
hspace	horizontální okraj	v pixelech
align	zarovnání obrázků	left, right, top, middle, absmiddle, baseline, bottom, absbottom
usemap	klikací mapa	jméno mapy nebo url mapy

V současné době popularity CSS stylů se doporučují atributy *width*, *height* a *border* nahradit stejnojmennými vlastnostmi CSS stylu, místo atributů *vspace* a *hspace* se doporučuje použít CSS vlastnost *margin* a zarovnání *align* je vhodné nahradit CSS vlastností *float* popřípadě *vertical-align*.

```
<body>
  <img src="obrázek.jpg" alt="alternativní popis obrázku" width="50" height="50" hspace="5" />

</body>
```

Obr. 2.7 Zdrojový kód pro zobrazení obrázku

Seznamy

Seznamy jsou populární položkou každého textového editoru. Je zřejmé, že i informace na webových stránkách je potřeba občas seřadit. K tomu obecně slouží číslované nebo odrážkové seznamy, přesně v takové podobě, jako je známe např. v Microsoft Wordu. Pro číslovaný seznam máme přiřazen termín uspořádaný seznam (angl. *ordered list*), ze kterého vychází název elementu <ol>, ekvivalentem odrážkového seznamu je neuspořádaný seznam (angl. *unordered list*), ze kterého vychází název elementu <ul>. Pro položku seznamu (angl. *list item*) se používá element

Tab. 2.8 Elementy pro vytváření seznamu

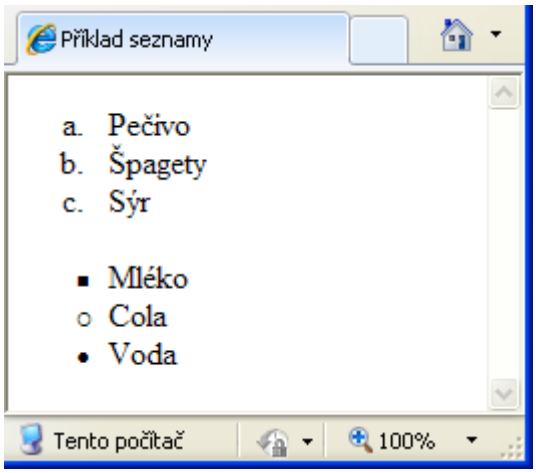
Element	Význam
li	položka seznamu
ol	uspořádaný seznam
ul	neuspořádaný seznam
dir	druh seznamu – již se nepoužívá
menu	druh seznamu – již se nepoužívá
dl	seznam termínů
dt	definovaný termín
dd	definice termínu

Tab. 2.9 Atributy elementu

Atribut	Význam	Hodnoty
Type	Druh odrážky	Disc, circle, square
	Druh číslování	1, A, a, I, i

```
<ol>
        <li type="a"> Pečivo</li>
        <li type="a"> Špagety</li>
        <li type="a"> Sýr </li>
</ol>
```

Obr. 2.8 Zdrojový kód pro uspořádaný seznam



Obr. 2.9 Zobrazení uspořádaného a neuspořádaného seznamu

Korespondenční úkol – neuspořádaný seznam

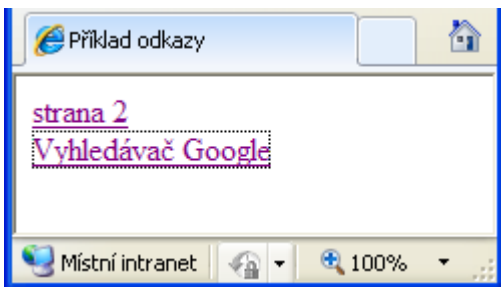
Vytvořte zdrojový kód pro neuspořádaný seznam který je zobrazen na obrázku 2.9.

Odkazy

Odkazy jsou v podstatě nejdůležitější částí webových stránek. Povinným atributem je zde cíl odkazu, který je zadán URL adresou. Může se jednat o stránku ve stejném adresáři webové aplikace, nebo o vzdálený link odkazující se na adresu jiného serveru. Důležitým atributem je *target* neboli cíl, který určí jestli se odkazující link otevře do nového resp. stejného okna prohlížeče.

```
<body>

 <a href="HTMLPage2.htm" target="_self"> strana 2</a> <br />
 <a href="http://www.google.com" target="_blank"> Vyhledávač Google</a>

</body>
```

Obr. 2.10 Zdrojový kód pro odkazy



Obr. 2.11 Zobrazení obou odkazů z předchozího příkladu

Pohledem na atributy elementu <a> vidíme atribut *name* sloužící po pojmenování nějakého úseku stránky, kterému říkáme záložka. Pokud tedy použijeme tento mechanismus, můžeme použít element <a> i pro odkazy uvnitř jedné stránky. Tento mechanismus je oblíben u online manuálů, které v horní části stránky uvedou obsah s relativními odkazy na kapitoly, které jsou psány zhora dolů za sebou a link na záložky pomocí atributu *name* tak umožní snadný pohyb uvnitř rozsáhlého textu.

Tab. 2.10 Atributy elementu <a>

Atribut	Význam	Hodnoty
href	cíl odkazu	URL odkazu
name	jméno záložky	pojmenování
target	cílový rám	jméno rámu, _blank, _top, _parent, _self
rel	druh odkazu	nofollow – google nebere v úvahu, nevyužíví se

Ke kapitole 2 je přiřazena demonstrační animace č. 1

Animace č. 1 je věnována tvorbě HTML dokumentu z dostupných elementů probraných v této kapitole.

Shrnutí kapitoly

HTML elementy mohou ve své definici používat buďto pouze název elementu v ostrých závorkách, nebo můžeme použít atributy elementů, které danému elementu určí ještě dodatečné vlastnosti. Těmito vlastnostmi bývá zarovnávání, velikost, barva a mnoho dalších. Syntaxe pak bude následující:

```
<název elementu atribut="hodnota" atribut="hodnota" >
```

V Html dokumentech můžeme používat barvy pro běžné písmo, nadpisy, pozadí stránky, odstíny buněk tabulky a další prvky. Barvy jsou vyjádřeny ve formátu RGB (*RedGreenBlue*) pomocí hexadecimálního kódu nebo přímo čísel odstínu od 0 do 255. Při použití hexadecimálního kódu použijeme před číselným vyjádřením barvy znak **#**. Definovat barvu tedy můžeme takto:

```
<body bgcolor="#C0C0C0" text="rgb(0,0,255)">
```

Hlavička HTML dokumentu je nezbytnou částí každé stránky. Je uvozena elementy <head></head>.Mezi těmito elementy se může nacházet pouze několik elementů: <title>, <link>, <base>, <meta>, <style>,<script> a jejich párové ukončení.

Použití jednotlivých elementů pro fyzické a logické formátování textu, použití tabulek, obrázků, seznamů a odkazů je přehledně uvedeno v tabulkách této kapitoly.

Úkol k řešení 2.1 – Obrázky jako linky (odkazy)

Vytvořte jednoduchou HTML stránku, ve které budou hrát obrázky roli odkazů na různé stránky, které se budou otevírat v novém okně.

Úkol k řešení 2.2 – Vytvoření jednoduché HTML stránky

Pomocí známých HTML elementů vytvořte stránku, ve které budete mít nadpisy kapitol dvou úrovní a použijete zanořený seznam:

- Jídlo
  - pečivo
  - máslo
  - vejce
  - sýr
- Pití
  - mléko
  - minerálka
  - víno

Úkol k řešení 2.3 – Horní a dolní index

Pomocí vhodných HTML elementů zajistěte zobrazení rovnice $x_1 + y^2 = z_1$

Úkol k řešení 2.4 – Tabulka

Vytvořte následující tabulku se sloučenými buňkami:

<table>
<tr><td>1</td><td>2</td><td>3</td><td>4</td></tr>
<tr><td colspan="3">5</td><td>6</td><td rowspan="2">7</td></tr>
<tr><td>8</td><td>9</td><td>10</td></tr>
</table>

Úkol k řešení 2.5 – link uvnitř dokumentu

Vytvořte jednu stránku s dlouhým textem rozčleněným do kapitol. Pomocí atributu *name* elementu `<a>` zajistěte snadný pohyb po stránce.

Kontrolní otázka 2.1

Jakým způsobem a k čemu se používají atributy HTML elementů?

Kontrolní otázka 2.2

Kde mohu do zdrojového kódu umístit informace o obsahu a autorovi stránek?

Kontrolní otázka 2.3

Jakým způsobem používáme barvy v HTML dokumentech?

Kontrolní otázka 2.4

Jakým zápisem můžeme obarvit písmo v dokumentu na žlutou barvu?

Kontrolní otázka 2.5

Jak zajistíme, že se odkaz neotevře do stejného okna, ze kterého je otevírán?

Kontrolní otázka 2.6

Proč se nedoporučuje používat element <u>?

Kontrolní otázka 2.7

Jaké jsou důvody pro nedoporučování starších tagů?

Kontrolní otázka 2.8

Jak zajistím, aby obsah buňky byl vycentrovaný na střed, když sousední buňka má evidentně více řádků?

Kontrolní otázka 2.9

Jakým způsobem slučujeme buňky tabulky?

Kontrolní otázka 2.10

Jakým způsobem se odkazuji elementem <a> uvnitř jednotlivých kapitol na jedné stránce?

3. KASKÁDOVÉ STYLY

Čas ke studiu: 2 hodiny

Cíl Po prostudování této kapitoly budete umět

- vysvětlit výhody použití CSS stylů
- používat kaskádové styly pro formátování stránek
- definovat všechny způsoby, jakým lze CSS styl přiřadit ke stránce
- vytvořit jednotný CSS styl pro celou webovou aplikaci
- definovat a správně používat syntaxi CSS stylu
- výhodně využívat selektory CSS stylů pro pokročilé formátování

Výklad

V rychlém sledu kurzu tvorby webu přistoupíme zase o kousek dále. Po vysvětlení principů použití HTML elementů je na řadě další pomůcka, a to jsou CSS styly. Tyto styly mnohdy nahradí složité formátování HTML elementů a umí toho také daleko více než atributy HTML elementů. Nejprve si řekneme proč je výhodné tyto styly používat, poté si ukážeme jakým způsobem a uvedeme opět syntaxi doplněnou o názorné příklady.

3.1 Důvody pro použití kaskádových stylů

I když bychom důvodů pro použití CSS stylů našli mnoho, jejich společným sjednocením by nakonec zůstali tyto tři hlavní důvody:

- Použití CSS stylů ušetří čas.
- Stránky, které používají CSS styly mají jednotný vzhled.
- Vzniká nám přehledný a dobře citovatelný zdrojový kód.

První důvod oceníme jak při tvorbě jedné či několika stránek, tak i při tvorbě složitějšího webového portálu. Jistě se atributy HTML elementů opakují a samozřejmě z druhého uvedeného důvodu jsou často totožné. Proto je výhodnější vlastnosti elementů nadefinovat jednou hromadně a pak se na toto formátování odkazovat. Dokážeme si jistě představit tu dlouhou práci při editaci stránek, pokud bychom chtěli drobně upravit jejich vzhled. Editace atributů stovek elementů by mohla zabrat tolik času jako samotný návrh nového webu. Při změně barev, formátování, pozicování, velikosti a dalších atributů nám tedy stačí editovat pouze styl, který se aplikuje na jednotlivé elementy ve všech stránkách aplikace. Stránky, které jsou součástí jednoho webového sídla musí mít vzhled jednotný a za použití CSS stylů máme také přehledný zdrojový kód, který můžeme v případě nutnosti lépe editovat.

Pojem k zapamatování: CSS

Zkratka CSS znamená *Cascading Style Sheets*, neboli doslovně šablony kaskádových stylů. Ve zkratce říkáme pouze kaskádové styly. Tyto styly dokáží pokročile formátovat elementy jazyka HTML.

3.2 Způsoby používání kaskádových stylů

Než přejdeme k syntaxi samotného CSS stylu, musíme vědět jakým způsobem vůbec styl do Html dokumentu dostat. Existují tři způsoby zápisu. Prvním z nich je tzv. *Inline zápis.* V podstatě CSS styl je atributem *style* Html elementu. Tímto způsobem si nijak nepomůže v šetření času, místem a přehledností, avšak jsou důvody, proč tento způsob používat. Jedním z nich je fakt, že inline zápis stylu dostane přednost před stylem v hlavičce. Nechcemeli tedy příslušný element formátovat podle hlavního stylu, můžeme to takto jednoduše vyřešit. Druhý způsob zápisu je tzv. *Interní zápis*. Tento zápis stylu se definuje v elementu <*head*>. Všechny použité elementy v html dokumentu pak získají uvedené vlastnosti definovaného stylu (pokud tedy nejsou vyjímky zajištěné jinak). Na obrázku 3.1 vidíme dva způsoby zápisu CSS stylu.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>CSS Styly</title>
    <style type="text/css">
        h2 {color:Blue; text-align:center;}
        h3 {color:Gray; text-align:left;}
        p  {color:navy;}
    </style>
</head>
<body>
<h1> Formátování webové stránky</h1>
<h2> CSS Styly</h2>
<p style="text-align:justify; color:Silver">
První možností je inline zápis přímo do atributu html elementu
</p>
<p> Druhou možností je Interní zápis, který je v hlavičce html stránky</p>
<p> Třetí možností je externí CSS styl, kterů se nachází v externím souboru, ten se
nejčastěji používá pro formátování více stránek najednou.</p>
 </body>
</html>
```

Obr. 3.1 Způsoby používání CSS stylu

Většinou však pokročilé webové stránky, portály a sídla neobsahují pouze jednu zdrojovou stránku. Musíme tedy zajistit, aby definovaný styl naformátoval elementy ve všech zdrojových stránkách portálu. Nejčastěji se tak volí *Externí zápis* kaskádového stylu. Obvykle je externí soubor s kaskádovým stylem ve stejném adresáři, či v nějakém podadresáři zdrojové aplikace a samotný odkaz na styl je proveden v každé zdrojové stránce v hlavičce za pomocí elementu <*link*>.
Na obrázku 3.2 vidíme uvedené skutečnosti, za prvé odkaz na styl v hlavním dokumentu a za druhé zdrojový kód kaskádového stylu v externím souboru. Vidíme, že obsahem externího souboru je pouze definice stylu bez jakýchkoliv dalších informací.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>CSS Styly</title>
   <link rel="Stylesheet" type="text/css" href="MujStyl.css" />
</head>
<body>
```

MujStyl.css* | Styly.htm | Default.htm* | Start Page

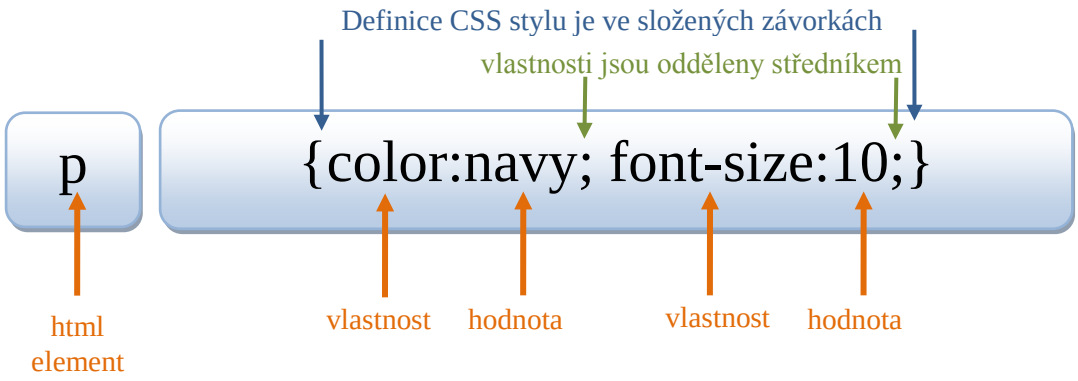
```
h2 {color:Blue; text-align:center;}
h3 {color:Gray; text-align:left;}
p  {color:navy;}
```

Obr. 3.2 Použití externí definice CSS stylu (nahoře odkaz v sekci <head>, dole již vlastní styl)

3.3 Syntaxe formátování CSS

Obecně platí, že formátování pomocí kaskádových stylů je trochu „upovídanější“ než u atributů Html elementů. Např. zarovnání písma je u html atributu pomocí klíčového slova ***align***, u CSS je to ***text-align***. Proto také mnozí uživatelé dají přednost atributům elementů html před CSS styly, což je ale chybné. Zpočátku nepříliš pěkná syntaxe se ve výsledném dokumentu rozhodně vyplatí a opět budou platit zmiňované tři důvody proč používat CSS.

Syntaxe je následující: nejprve následuje element, pro který máme připraven formátovací styl. Poté následují ve složených závorkách vlastnosti elementů a jejich hodnoty za dvojtečkou, jednotlivé vlastnosti jsou odděleny středníkem. Pokud budeme mít stejný formátovací styl pro více elementů, můžeme je uvést za sebou a oddělit čárkou.



Obr. 3.3 Syntaxe použití kaskádového stylu

Pojem k zapamatování: vlastnosti elementů při formátování pomocí CSS

Atributy HTML elementů se zapisují jinak než vlastnosti CSS stylů, přestože vyjadřují mnohdy stejnou funkčnost viz.(*align* vs. text-*align*).

Obecně platí, že CSS styly mají daleko širší možnosti použití než atributy HTML elementů.

3.4 Selektory kaskádových stylů

Při používání kaskádových stylů vznikla potřeba definovat nějaký styl s více vlastnostmi, který budeme aplikovat na určitý typ elementu. Tento styl je vhodné pojmenovat, neboli přiřadit jednoznačný identifikátor tzv. *id*, na které se pak budeme odkazovat. Id tak bude atributem html elementu.

```
<style type="text/css">
#zelenyText
{
text-align:center;
color:green;
}
</style>
</head>

<body>
<p id="zelenyText">Tento odstavec bude určitě zelený a centrovaný na střed</p>
<p>Tento odstavec se neformátuje.</p>
</body>
```

Obr. 3.4 Použití selektoru ***id*** kaskádového stylu

Druhým selektorem je selektor třídy, který na rozdíl od selektoru *id* aplikuje formát CSS stylu na více elementů. Selektor třídy se používá jako atribut *class* html elementu a před definicí stylu jeho název předchází tečka. Oba selektory *id* i *class* ohraničují své vlastnosti a hodnoty ve složených závorkách.

```
<html>
<head><title> CSS styly</title>
<style type="text/css">
.podtrzenyText
{
text-align:center;
text-decoration:underline;
}
</style>
</head>

<body>
<h1 class="podtrzenyText">Tento nadpis bude potržený</h1>
<p> Tento odstavec se neformátuje </p>
<p class="podtrzenyText">Podtržený bude také tento odstavec.</p>
</body>
</html>
```

Obr. 3.5 Použití selektoru ***class*** pro definici více elementů za použití kaskádového stylu

3.5 Ukázka použití kaskádových stylů

V této kapitole se konečně dostáváme ke konkrétnímu formátování pomocí kaskádových stylů. Jednotlivé vlastnosti jsou podrobně uvedeny v tabulce a demonstrovány na příkladech, které jsou v prostředí Visual Studia spuštěny. Jednotlivé příklady na obrázcích tak zobrazují definici CSS stylů, jejich aplikaci na elementy Html a výsledný efekt v prohlížeči (ten je simulován oknem design v prostředí visual studia, stránky tak není nutné spouštět v prohlížeči, výsledek je vidět v návrhovém okně). V příkladech si probereme nejčastěji používané formátovací možnosti elementů, jejich celkový výčet je samozřejmě nad rámec tohoto výukového textu. Avšak důležité je pochopit princip formátování pomocí CSS, což tyto příklady bezesporu splňují.

První kategorií použití CSS stylu se obecně týká fontu. Mnozí uživatelé stále neznají definici rodin písma *Serif a Sans-serif*, což znamená bezpatkové a patkové písmo, a proto si je na obrázku předvedeme.

Pojem k zapamatování: Serif versus Sans-Serif

Serif znamená patkové písmo, Sans-serif bezpatkové. Typickým představitelem bezpatkového písma je Arial, zatímco patkovým písmem je Times New Roman.

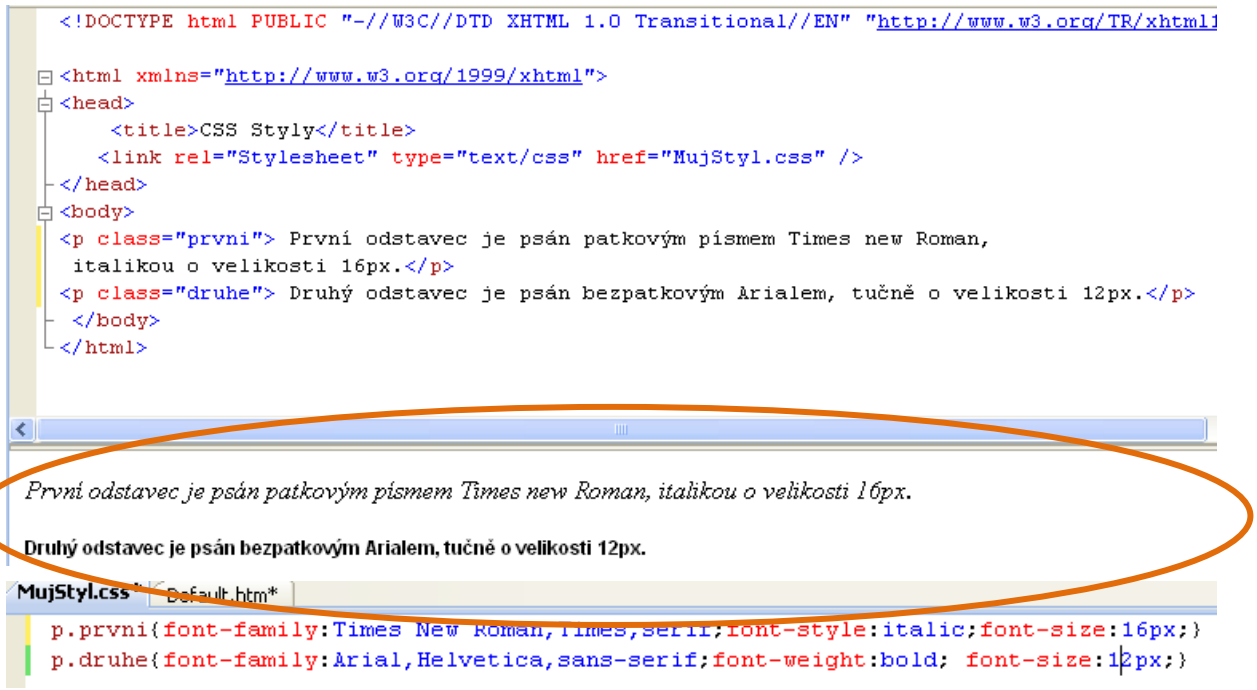
Obr. 3.6 Rozdíl mezi bezpatkovým (vlevo) a patkovým písmem

Pro nastavení fontu můžeme používat pět základních vlastností CSS, které jsou uvedeny v tabulce 3.1.

Tab. 3.1 Nastavení fontu pomocí CSS

Vlastnost	Popis	Hodnoty
font-family	Font písma a jeho typ	Seznam písem (patkové, bezp.)
font-style	Normální, kurzíva, skloněné	normal, italic, obligue
font-size	Velikost písma – slovně,výškou v pixelech nebo procentuálně	xx-small, x-small, small, medium. Large, x-large, xx-large, smaller, larger, *počet* px, %
font-weight	Hrubost písma – normální tučné, tučnější, světlejší, vyjádřené číslem	normal, bold, bolder, ligter, 100…900
font-variant	Kapitálky nebo normální	small-caps, normal
font	Obecně cokoliv z výše definovaných vlastností	

Po výčtu možností nastavení fontu je vhodné uvedené vlastnosti demonstrovat na příkladu. Na následujícím obrázku vidíme zdrojový kód dvou odstavců s odpovídajícím výstupem v prohlížeči. Pod tímto výstupem je vyobrazen obsah kaskádového stylu obou odstavců.



Obr. 3.7 Příklad nastavení fontu pomocí CSS

Vlastnosti fontu plně nahrazují nevhodný element `<font>` , který se nedoporučuje používat. Další nedoporučované html elementy, které jsme si představili v předchozí kapitole, se týkaly podtržení a přeškrtnutí písma. Tyto vlastnosti textu, pokud je potřebujeme použít, se nahrazují vlastností CSS stylu *text-decoration*. Tyto a také další vlastnosti písma, z nichž nejčastěji používáme definici barvy a zarovnávání, jsou uvedeny v tabulce 3.2.

Tab. 3.2 Nastavení formátu textu pomocí CSS

Vlastnost	Popis	Hodnoty
color	barva písma slovně, rgb číslem nebo hexa hodnotou	color_name, #ffffff nebo rgb(255,255,255)
direction	směr písma	ltr, rtl

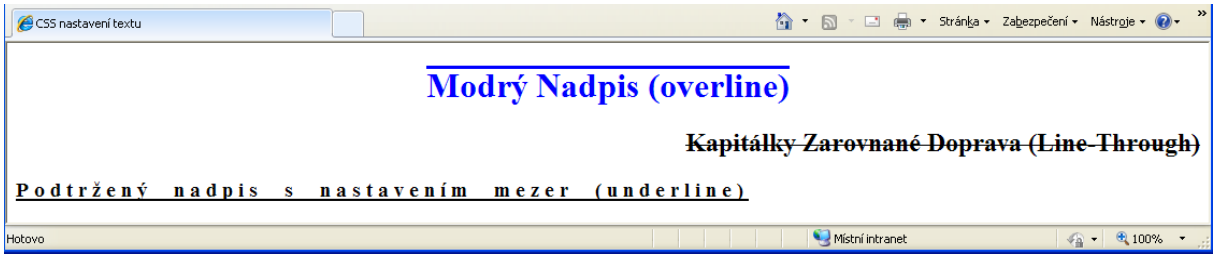
line-height	vzálenost mezi řádky	*číslo*, *délka*, *%* nebo normal
letter-spacing	mezera mezi písmeny	*délka* nebo normal
text-align	zarovnání textu	Left, right, center,justify
text-decoration	dekorace - podtržení, nadtržení, přeškrknutí, blikání	none, underline, overline, line-through, blink
text-intent	odsazení prvního řádku	*délka* nebo *%*
text-transform	nastavení písma v textu na všechna velká, malá nebo jen začátky slov	none, capitalize, uppercase, lowecase
vertical-align	vertikální zarovnání	baseline, sub, super, top, text-top, middle, bottom, text-bottom, *délka* a %
white-space	zalamování konce řádku	normal, pre, nowrap
word-spacing	mezera mezi slovy	*délka* nebo normal

Z uvedené tabulky si na příkladu ukážeme použití některých vlastností. Nadefinujeme si tři html elementy a nastavíme u nich zmíněné podtržení, přeškrtnutí, zvolíme barvu, zarovnávání a ukážeme si také nastavení mezer mezi písmeny a slovy. Pro písmo elementu *h2* zvolíme kapitálky.

```
<style type="text/css">
h1 {text-decoration:overline;
    color:Blue;
    text-align:center;
    }
h2 {text-decoration:line-through;
    text-align:right;
    text-transform:capitalize;}
h3 {text-decoration:underline;
    letter-spacing:5px;
    word-spacing:10px;
    }
</style>
</head>

<body>
<h1>Modrý Nadpis  (overline)</h1>
<h2>Kapitálky zarovnané doprava (line-through)</h2>
<h3>Podtržený nadpis s nastavením mezer (underline)</h3>
</body>
```

Obr. 3.8 Zdrojový kód CSS vlastností nastavení formátu textu

Výsledné nastavení tří nadpisů pomocí CSS stylu vidíme v okně prohlížeče na obrázku 3.9.



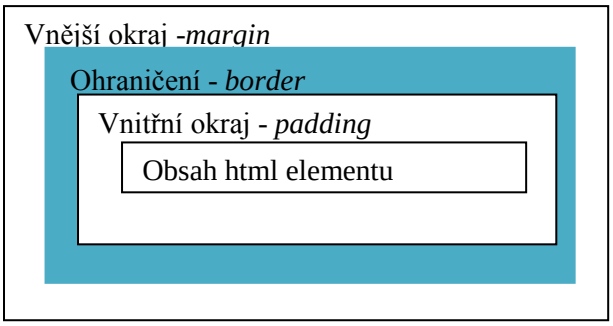
Obr. 3.9 Formátování elementu podle zdrojového kódu z obr. 3.8

Další obecně často používanou vlastností je pozadí elementu. Jako pozadí nějakého elementu můžeme zvolit barvu nebo obrázek. Barvu opět definuje třemi možnostmi: jménem, rgb čísly nebo hexa-kódem. Obrázek je nejčastěji ve formátech gif, png nebo jpg. Obrázky se většinou používají ve spojitosti s těmito CSS vlastnostmi jako textury, loga nebo proužky pro grafický návrh stránek, nemáme teď na mysli klasický obrázek, který chceme uživatelům ukázat. U obrázku je nutné počítat s obtížnou prací díky jeho rozměrům. Nemůžeme obrázek nějak deformovat či roztahovat změnou jeho proporcí. Máme ale některé možnosti fixace, uchycení, skrolování či opakování. Tyto možnosti jsou uvedeny v tabulce 3.3

Tab. 3.3 Pozadí elementu

Vlastnost	Popis	Hodnoty
background-color	barva pozadí elementu	*rgb*barva, *hex*barva, *jméno*barva, transparent
background-image	obrázek na pozadí	none, url(adresa)
background-repeat	opakování obrázku pozadí	repeat, no-repeat, repeat-x, repeat-y
background-attachment	fixace nebo scrolling pozice pozadí	scroll, fixed, inherit
background-position	poloha obrázku pozadí	left top, left center, left bottom, right top, right center, right bottom, center top, center center, center bottom, *x% y%*, *xpos ypos*, inherit
background	nastavení všech vlastností v jedné deklaraci	background-color, background-image, background-repeat, background-attachment, background-position

Nejen u obrázků, ale obecně u mnoha elementů, nám při definici plochy dělá problém nastavení počátku elementu. Abychom neměli např. písmo natlačené příliš na okraj stránky, použijeme okraje tzv. box modelu html elementu. Tento box model určuje plochy kolem html elementu, které lze pomocí vlastností CSS nastavit na požadovanou hodnotu.



Obr. 3.10 Box model html elementu

Tab. 3.4 Box Model návrhu a rozložení (tzv. *design* & *layout*)

Vlastnost	Popis	Hodnoty
margin	Nastavení šířky vnějšího okraje – všechny čtyři okraje v jedné deklaraci	margin-top, margin-left, margin-bottom, margin-right
margin-top	horní vnější okraj	auto, délka, %
margin-left	levý vnější okraj	auto, délka, %

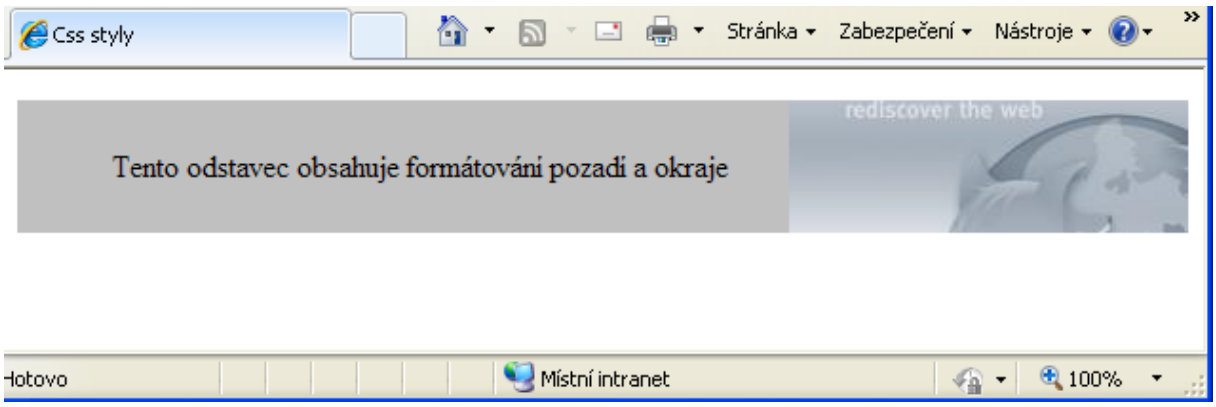
margin-bottom	spodní vnější okraj	auto, délka, %
margin-right	pravý vnější okraj	auto, délka, %
padding	nastavení šířky vnitřního okraje – všechny čtyři okraje v jedné deklaraci	padding-top, padding-left, padding-bottom, padding-right
padding-top	horní vnitřní okraj	délka, %
padding-left	levý vnitřní okraj	délka, %
padding-bottom	spodní vnitřní okraj	délka, %
padding-right	pravý vnitřní okraj	délka, %

Na praktickém příkladu si ukážeme nastavení obrázku na pozadí elementu a také vnější okraj – *margin*. Obrázek nebudeme na pozadí opakovat a umístíme jej doprava. Element odstavec bude mít ještě nastavenou barvu pozadí a hodnoty vnějších okrajů.

```
<style type="text/css">
p
{
background-color:silver;
background-image:url(obrázek.jpg);
background-repeat:no-repeat;
background-position:right;
}
p.padding
{
padding-top:25px;
padding-bottom:25px;
padding-right:50px;
padding-left:50px;
}
</style>
</head>
<body>
<p class="padding">Tento odstavec obsahuje formátování pozadí a okraje</p>
</body>
```

Obr. 3.11 Zdrojový kód CSS formátování pozadí a okraje

Vlastnosti odstavce jsou dle zdrojového kódu z obr. 3.11 deklarovány jako CSS vlastnosti elementu, ale i pomocí selektoru *class* s tečkovou konvencí. Výsledný vzhled odstavce v prohlížeči vidíme na obrázku 3.12.



Obr. 3.12 Zobrazení formátování odstavce podle zdrojového kódu z obrázku 3.11

Rovněž při nastavení proporcí je vhodné místo atributů html elementů použít CSS vlastnosti. Nastavení se provádí obvykle v pixelech nebo procentuálně. Seznam vlastností pro nastavení proporcí elementů je uveden v tabulce 3.5.

Tab. 3.5 Velikost elementu

Vlastnost	Popis	Hodnoty
width	nastavení šířky elementu	auto, *šířka* v px, procento
height	nastavení výšky elementu	auto, *výška* v px, procento
min-width	nastavení min. šířky elementu	*šířka* v px, procento, inherit
max-width	nastavení max. šířky elementu	none, *šířka* v px, procento, inherit
min-height	nastavení min. výšky elementu	*výška* v px, procento, inherit
max-height	nastavení max. výšky elementu	none, *výška* v px, procento, inherit

Nastavení proporcí html elementu si ukážeme na příkladu nastavení rozměru obrázku. Na obrázku 3.13 je uveden zdrojový kód s definicí tří velikosti pomocí selektoru class, na obrázku 3.14 je výsledný příklad zobrazen v prohlížeči.

```
<style type="text/css">
img.velky
{
height:auto;
}
img.malinky
{
height:20px;
}
img.vlastni
{
height:50px;
width:60px;
</style>
</head>
<body>
<img alt="opera" class="velky" src="opera.jpg"  />
<img alt="opera" class="malinky" src="opera.jpg"  />
<img alt="opera" class="vlastni" src="opera.jpg" />
</body>
```

Obr. 3.13 Zdrojový kód CSS nastavení velikosti elementu

Obr. 3.14 CSS nastavení velikosti elementu podle zdrojového kódu z obr. 3.13

Tato kapitola neobsahuje kompletní seznam CSS vlastností, ale vybírá pouze ty nejdůležitější. Seznam dalších elementů najdeme ve W3C specifikaci nebo na portálech zabývajícími se kaskádovými styly.

Seznam vybraných portálu s popisem CSS vlastností

1. http://www.w3.org/Style/CSS/
2. http://www.jakpsatweb.cz/css/css-vlastnosti-hodnoty-prehled.html
3. http://www.w3schools.com/css/css_reference.asp

Korespondenční úkol – barevné rozlišení navštívených odkazů

Pomocí CSS se dají rozlišit již navštívené odkazy od ještě nenavštívených. V praxi se tato vlastnost hojně využívá. Vyhledejte a použijte příslušnou vlastnost.

Ke kapitole 3 je přiřazena demonstrační animace č.2

První část animace č. 2 obsahuje praktickou ukázku použití CSS stylů probraných v této kapitole. Další části animace jsou navazující ukázka tvorby layoutu pomocí CSS, podrobně probrané v následující kapitole.

Shrnutí kapitoly

Zkratka CSS znamená *Cascading Style Sheets*, neboli doslovně šablony kaskádových stylů. Ve zkratce říkáme pouze kaskádové styly. Tyto styly dokáží pokročile formátovat elementy jazyka HTML. Existují tři hlavní důvody pro použití kaskádových stylů:

1. Použití CSS stylů ušetří čas.
2. Stránky, které používají CSS styly mají jednotný vzhled.
3. Vzniká nám přehledný a dobře citovatelný zdrojový kód.

Rovněž tři jsou způsoby použití zápisu CSS stylu. Prvním z nich je tzv. *Inline zápis.* V postatě CSS style je atributem *style* Html elementu. Druhý způsob zápisu je tzv. *Interní zápis*. Tento zápis stylu se definuje v elementu <*head*>. Všechny použité elementy v html dokumentu pak získají uvedené vlastnosti definovaného stylu. Třetím a nejčastějším způsobem zápisu je *Externí zápis* kaskádového stylu. Obvykle je externí soubor s kaskádovým stylem ve stejném adresáři, či v nějakém podadresáři zdrojové aplikace a samotný odkaz na styl je proveden v každé zdrojové stránce v hlavičce za pomocí elementu <*link*>.

Syntaxe CSS formátování je následující: nejprve následuje element, pro který máme připraven formátovací styl. Poté následují ve složených závorkách vlastnosti elementů a jejich hodnoty za dvojtečkou, jednotlivé vlastnosti jsou odděleny středníkem. Pokud budeme mít stejný formátovací styl pro více elementů, můžeme je uvést za sebou a oddělit čárkou.

Atributy HTML elementů se zapisují jinak než vlastnosti CSS stylů, přesto, že vyjadřují mnohdy stejnou funkčnost viz.(*align* vs. text-*align*). Při používání kaskádových stylů vznikla potřeba definovat nějaký styl s více vlastnostmi, který budeme aplikovat na určitý

typ elementu. Tento styl je vhodné pojmenovat, neboli přiřadit jednoznačný identifikátor tzv. *id*, na které se pak budeme odkazovat. Id tak bude atributem html elementu. Druhým selektorem je selektor třídy, který na rozdíl od selektoru *id* aplikuje formát CSS stylu na více elementů. Selektor třídy se používá jako atribut *class* html elementu a před definicí stylu jeho název předchází tečka. Oba selektory *id* i *class* ohraničují své vlastnosti a hodnoty ve složených závorkách.

Pro formátování fontu je důležitou volbou typ rodiny písma. Serif znamená patkové písmo, Sans-serif bezpatkové. Typickým představitelem bezpatkového písma je Arial, zatímco patkovým písmem je Times New Roman.

V této kapitole jsme představili nejpoužívanější CSS vlastnosti html elementů pro nastavení fontů, formátování písma, nastavení barev, proporcí, pozadí a další.

Nejen u obrázků, ale obecně u mnoho elementů při definici plochy dělá problém nastavení počátku elementu. Abychom neměli element natlačený příliš na okraj stránky, použijeme okraje tzv.box modelu html elementu. Tento box model určuje plochy kolem html elementu, které lze pomocí vlastností CSS nastavit na požadovanou hodnotu.

Tato kapitola neobsahuje, kompletní seznam CSS vlastností, ale vybírá pouze ty nejdůležitější. Seznam dalších elementů najdeme ve W3C specifikaci na adrese: http://www.w3.org/Style/CSS/

Úkol k řešení 3.1 – Definice kaskádových stylů

Vytvořte html stránku, kde použijete inline interní zápis definice kaskádového stylu. Pro procvičení zvolte některé vlastnosti formátování fontu, textu a barev.

Úkol k řešení 3.2 – Definice kaskádového stylu externím zápisem

Dle výukového textu vytvořte soubor s externím kaskádovým stylem, který využijete ve všech stránkách webové aplikace. Vytvořte tak několik html stránek, které budete jednotně formátovat z jednoho souboru s kaskádovým stylem.

Úkol k řešení 3.3 – Patkové vs. bezpatkové písmo

Vytvořte kaskádový styl, ve kterém uplatníte pro různé elementy oba typy písma, patkové i bezpatkové.

Úkol k řešení 3.4 – Ohrničení elementů v box modelu

Ve výukovém textu jsme si představili box model html elementu s vnějšími a vnitřními okraji. Nevěnovali jsme se ale ohraničení elementu – *border*. Vyhledejte příslušné CSS vlastnosti ohraničení a vyzkoušejte na vhodném příkladu.

Kontrolní otázka 3.1

Jaké jsou důvody pro použití kaskádových stylů?

Kontrolní otázka 3.2

Jakým způsobem můžeme vložit do html dokumentu kaskádový styl?

Kontrolní otázka 3.3

Jaký je rozdíl mezi tzv. fontem bezpatkového a patkového písma?

Kontrolní otázka 3.4

K čemu slouží ve formátování klíčové slovo *class*?

Kontrolní otázka 3.5

Jak budete postupovat, pokud budete chtít pomocí CSS stylu definovat vlastnosti elementu se kterým jste se v CSS stylech dosud nesetkali? (např. tvar odrážky nečíslovaného seznamu)

Kontrolní otázka 3.6

Co je to selektor CSS a k čemu se používá?

Kontrolní otázka 3.7

Jaký je rozdíl mezi atributem html elementu a vlastností CSS stylu?

4. LAYOUT WEBOVÝCH STRÁNEK A POZICOVÁNÍ

Čas ke studiu: 3 hodiny

Cíl Po prostudování této kapitoly budete umět

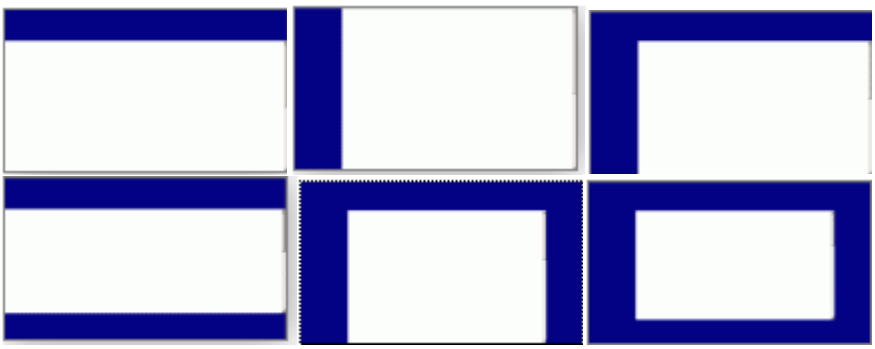
- vysvětlit a popsat různé formy layoutů
- správně zvolit layout pro své web stránky
- popsat způsoby, jakým můžeme layout vytvořit
- používat CSS vlastnosti layoutu a aplikovat je na html elementy
- vytvořit jednotný layout pomocí CSS stylu pro celou webovou aplikaci
- vyhledávat dostupné CSS layouty a aplikovat je do návrhu svých stránek
- vytvářet pokročilé CSS menu ve svých stránkách

Výklad

V předchozí kapitole jsme si vysvětlili podstatu formátování pomocí kaskádových stylů. Nyní přejdeme ještě o kousek dále a ukážeme si jakým způsobem využijeme zkušenosti z předchozích kapitol při návrhu stránky.

4.1 Různé formy layoutů

Při návrhu webových stránek musíme na samém počátku určit, jaký vizuální formát bude naše stránka mít. Oprostíme se nyní od grafických prvků, kterými jsou loga, obrázky, meníčka a různé další doplňky. Tato zdokonalení mají samozřejmě svou roli, ale musíme na samém počátku zvážit, jakým způsobem navrhneme samotný layout stránky neboli její pracovní rozvržení.



Obr 4.1 Příklady rozložení pracovní plochy webové stránky

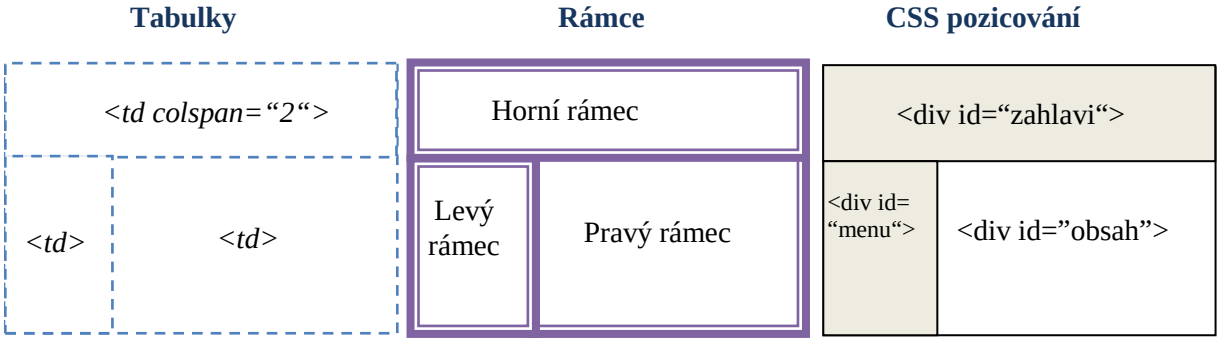
Podstatou návrhu je určení, k čemu stránky budou sloužit, jaký bude jejich informační obsah a jakou funkčnost mají plnit. Cílem tohoto učebního textu není profesionální design, ani grafické zpracování stránek, přesto musíme dbát určitých vhodných a zaběhnutých postupů. Především musíme určit, kde bude zobrazován hlavní informační obsah stránky, kde bude navigační lišta s odkazy, kolik dalších prvků na stránce budeme zobrazovat apod. Podle toho také můžeme volit vícesloupcový layout, nebo layout se záhlavím, zápatím apod. Další prvky pak budeme umísťovat do předem určených míst, kde budou ve všech stránkách aplikace, tzn. např. navigační lišta s odkazy v levém sloupci, logo v hlavičce stránky, letopočet s copyrightem v zápatí, aktuality v pravém sloupci a podobně.

Pojem k zapamatování: Layout webové stránky

Layout webové stránky znamená její rozvržení na obrazovce a je nejdůležitější částí návrhu web stránky. Před samotným vytvořením layoutu musíme udělat analýzu, jaké prvky budou stránky obsahovat, kde a jaký bude informační obsah stránek.

4.2 Tabulky, rámce a CSS styly

Rozložení obsahu stránek je řešeno různými způsoby, v různých dobách byly preferovány všechny. Nyní si je v krátkosti shrneme, neboť výhoda použití v dnešní době je jednoznačně na straně kaskádových stylů. Ale popořádku. Prvním způsobem jak můžeme vytvořit rozložení obsahu na stránku do různých částí obrazovky je použití tabulky. Pomocí spojování buněk a nastavení velikosti řádku se pohodlně určil rozměr požadovaných míst pro rozložení na stránce. Tohle řešení mělo ale jednu nevýhodu. Tabulka se zobrazí, až se celá načte do prohlížeče. To znamená, že uživatel musí čekat, až se načtou všechny elementy včetně obrázků a to při pomalejším internetovém připojení může zabrat nějaký čas, který by při postupném načtení stránky již využil k jejímu čtení. Tuto nevýhodu je možné obejít definicí více tabulek na stránce, ale pak se taková stránka obtížněji edituje.



Obr. 4.2 Tři způsoby návrhu layoutu

Druhou možností a velkolepou novinkou roku 1996 bylo zavedení rámců tzv. `<frame>`. Hlavní stránka neobsahovala element `<body>` ale definovala množinu rámů `<frameset>`. Množina rámů obsahuje již jednotlivé rámy `<frame>`, které ve svých atributech mají definovanou výšku a šířku. Ve výsledku pak máme navigační obsah v jednom rámu a veškerý informační obsah je směrován atributem *target* do hlavního rámu. Ukázkový příklad na rámy zde neuvádíme, neboť tuto techniku nedoporučuji vůbec používat. Rámy mají velmi mnoho nevýhod od problému s otevíráním odkazů v novém okně, přes špatnou funkčnost při tisku z důvodu jediného aktivního rámu, až po problémy s indexací vyhledávacích robotů. Třetí možností, která se dříve nepoužívala, protože dvě předchozí byly jednodušší, je pozicování pomocí CSS. V dnešní době již však většina moderních webů a portálů používá tento způsob. Je to dáno nejen rozvojem specifikace CSS, ale také neustále se vyvíjejícími komfortními nástroji pro práci s těmito technikami. V následující kapitole si však ukážeme, že ani ruční nastavení pozicování nemusí být problémem pro rychlé, pohodlné a fungující stránky založené na layoutu pomocí CSS. Na obrázku 4.2 vidíme všechny tři způsoby návrhu rozvržení informačního obsahu stránek.

Pojem k zapamatování: Doporučení pozicování layoutu webových stránek

V současné době je doporučováno navrhovat rozmístění objektů na stránce pomocí CSS pozicování. Tabulkový layout či dokonce rámy jsou zastaralými technikami. Prohlížeče je sice podporují, ale přesto způsobují menší či větší problémy při návrhu či zobrazení takto zpracovaných stránek.

4.3 CSS pozicování

Pozicování elementů html nám umožňuje umístit objekt na stránku doslova, kam chceme. Pozicovat můžeme absolutně nebo relativně. Z názvu termínu *absolutně* vyplývá, že daný objekt se bude nacházet přesně v souřadnicích, které mu určíme. *Relativně* naopak znamená posunutí v různých směrech oproti běžné poloze. V tabulce 4.1 jsou definovány CSS vlastnosti pro pozicování elementů. Všimněme si, že vlastnost *left* určuje posunutí doprava, můžeme si tuto vlastnost vysvětlit jako posunutí v pixelech z levého okraje stránky.

Tab. 4.1 CSS definice pozicování

Vlastnost	Popis	Hodnoty
position	umístění elementu	absolute, fixed, relative, static, inherit
left	posunutí doprava	auto, *délka* v px, *délka* v %, inherit
top	posunutí dolů	auto, *délka* v px, *délka* v %, inherit
right	pozicování od pravého okraje	auto, *délka* v px, *délka* v %, inherit
bottom	pozicování od spodního okraje	auto, *délka* v px, *délka* v %, inherit
clip	oříznutí absolutně pozicovaného elementu	shape, auto, inherit
overflow	přetékání nebo oříznutí elementu	auto, hiden, scroll, visible, inherit
z-index	definice překrývání elementu (v ose z)	auto, *číslo*
visibility	viditelnost elementu	visible, hidden

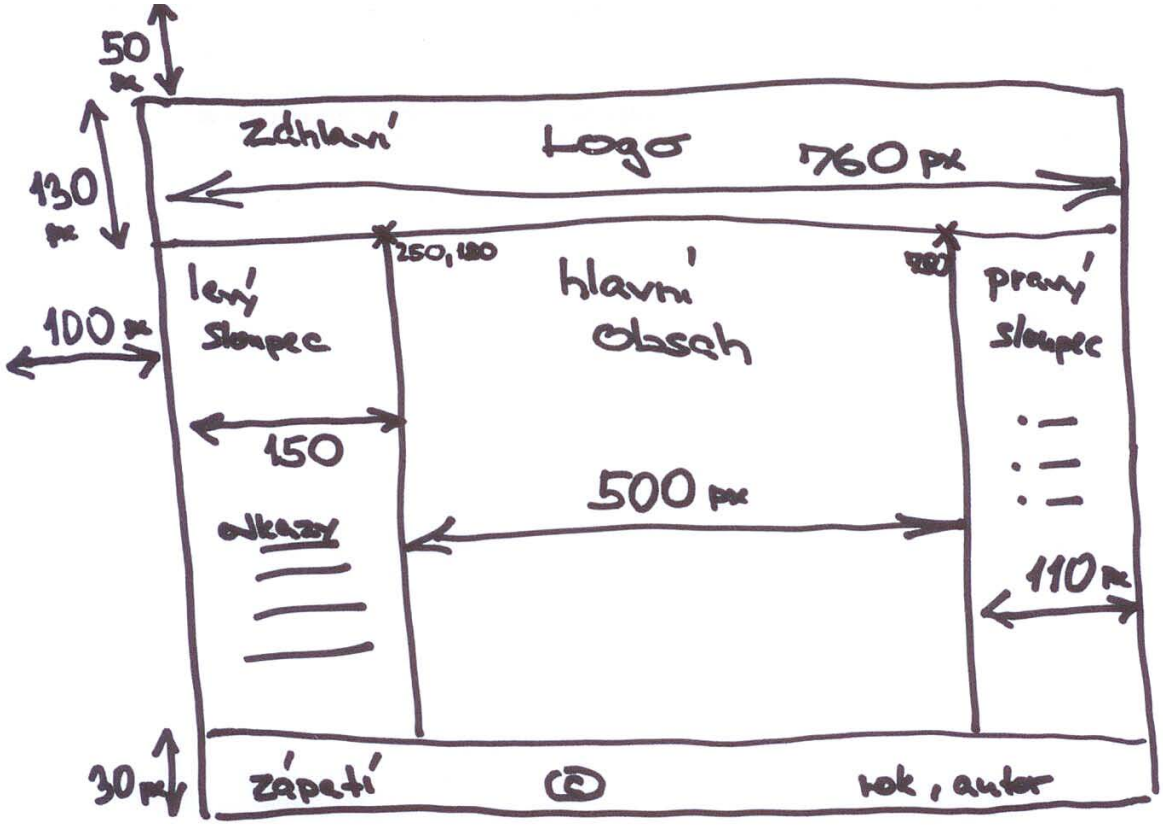
V souvislosti s pozicováním se můžeme zejména u obrázku setkat s problémem obtékání textem. Defaultně prvky obtékané nejsou, výchozí hodnotou je *none*. V tabulce 4.2 jsou obsaženy CSS vlastnosti pro obtékání elementu. Podmínkou možnosti obtékání je nastavení šířky obtékaného elementu. Vlastnost *clear* určuje vykreslování elementu až pod obtékanými prvky.

Tab. 4.2 CSS definice obtékání prvku

Vlastnost	Popis	Hodnoty
float	Umístění plovoucího objektu	left, right, none, inherit
clear	Nastavení, z kterého směru nebudou povoleny další plovoucí objekty	left, right, both, none, inherit

Nejvhodnějším vysvětlením problematiky pozicování je ukázka praktického příkladu. Navrhneme si layout webových stránek, který bude pozicován absolutně.Na obrázku 4.3 je představa vzhledu stránek zakreslena rukou na papír. Představa je taková že v levém sloupci bude navigační lišta s vertikálním menu, informační obsah se bude zobrazovat do největší části stránky pozicované uprostřed. Stránka bude obsahovat záhlaví, ve kterém bude umístěno logo stránek a zápatí pro informaci o autorovi, datu aktualizace apod. V pravém sloupci bude místo na aktuality a externí odkazy. Do navrženého layoutu byly zakresleny rozměry v pixelech.

Zvolíme absolutní pozicování, přičemž dodržíme základní vzdálenost 100 bodů od levého okraje a 50 bodů od horního okraje. Tyto údaje si musíme zapamatovat a při absolutním pozicování vždy u jednotlivých úseků přičíst k plánovaným rozměrům. Z toho plyne, že pokud levý sloupec má plánovanou šířku 150, proto hlavní sloupec bude mít vlastnost *left* rovnu 250, což odpovídá hodnotě 100 + 150 a vlastnost *top* rovnu 180, což odpovídá součtu výšky záhlaví a okraje stránky 130 + 50. Tímto způsobem budeme přičítat plánované okraje ke všem souřadnicím v příslušném směru.



Obr. 4.3 Ruční návrh stránky, kterou budeme realizovat

V externím souboru *pozicovani.css* nadefinujeme souřadnice absolutního pozicování. U jednotlivých částí layoutu ještě nastavíme dílší vlastnosti jako barva pozadí, písma apod. Výsledný obsah souboru se zdrojovým kódem vidíme na obrázku 4.4.

pozicovani.css

```
#hlavni    { position: absolute; width: 500px; top: 180px; left: 250px }
#zapati    {position: absolute; width: 760px; top: 520px; left: 100px; height: 30px}
#zahlavi   { position: absolute; width: 760px; top: 50px; left: 100px; height: 130px}
#vlevo     { position: absolute; width: 150px; top: 180px; left: 100px }
#vpravo    { position: absolute; width: 110px;height: 340px; top: 180px; left: 750px}
#vlevo     { background-color: silver; color: blue; border-style: none;
           border-width: medium; height: 340px }
#vlevo     { color: black }
#vpravo    { background-color: silver; color: white; border-style: none;
           border-width: medium }
#zahlavi, #zapati { background-color: navy; color: yellow;
                 border-style: none; border-width: medium }
div        { padding: 6px }
```

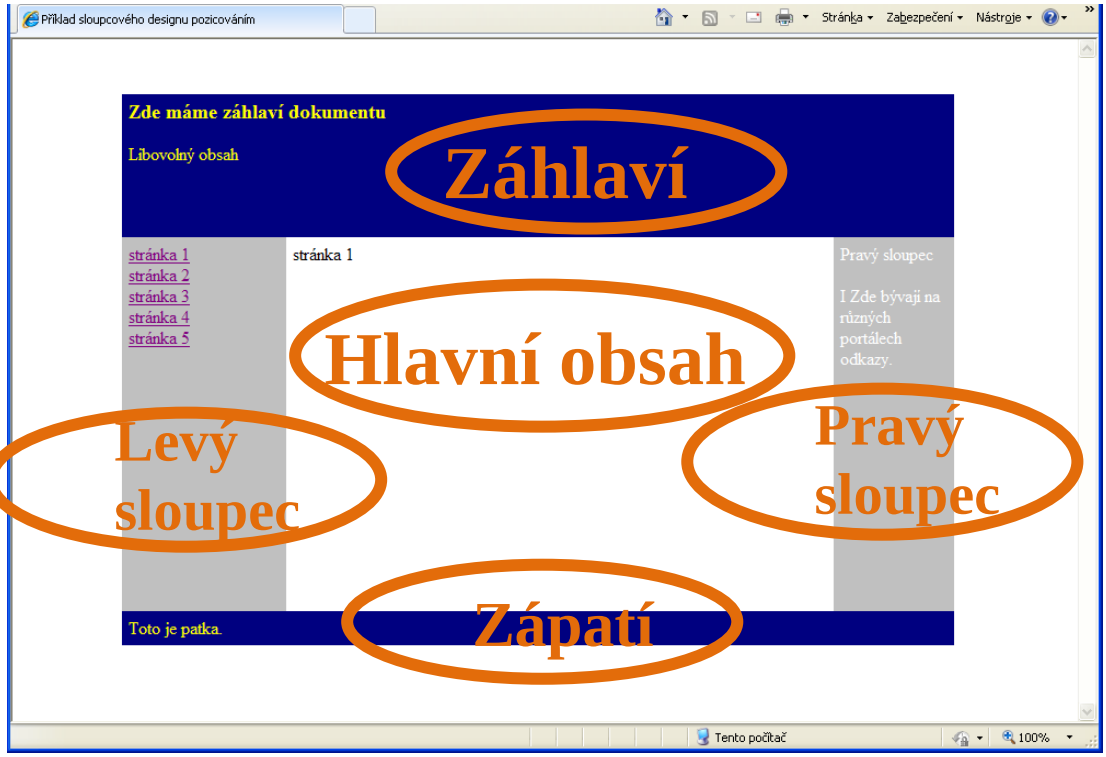
Obr. 4.4 pozicování layoutu stránky

V hlavním souboru již budeme pracovat pouze s elementy `<div>`, jejichž *id* se bude odkazovat na vlastnosti kaskádových stylů. Zdrojový kód html jednotlivých stránek tak bude přehledný a snadno editovatelný. Element `<div id="vlevo" >` bude obsahovat zmíněné menu s odkazy na další stránky, které budou s totožným layoutem, pouze s rozdílným informačním obsahem v elementu `<div id="hlavni" >`. Element `<div id="zahlavi" >` je určen pro zobrazení logo či názvu stránek v horní části stránky, tento element bude rovněž shodný pro všechny stránky, které je možné zobrazit z navigačního menu.

Na obrázku 4.5 vidíme přehledný zdrojový html kód jedné z navržených stránek. Výsledný layout stránek je zobrazen na obrázku 4.6. Jedná se o ukázkový příklad layoutu, stránky jsou tudíž prázdné, do jednotlivých částí jsme zapsali pouze informativní text.

```
<html>
<head>
<title>Příklad sloupcového designu pozicováním</title>
<link rel="stylesheet" type="text/css" href="pozicovani.css">
</head>
<body>

<div id="hlavni">
        stránka 1
</div>
<div id="zapati"><p>Toto je patka.</p>
</div>
<div id="zahlavi">
        <h3>Zde máme záhlaví dokumentu</h3>
        <p> Libovolný obsah </p>
        <p> 
</div>
<div id="vlevo">

<a href="1.htm"> stránka 1 </a><br>
<a href="2.htm"> stránka 2 </a><br>
<a href="3.htm"> stránka 3 </a><br>
<a href="4.htm"> stránka 4 </a><br>
<a href="5.htm"> stránka 5 </a><br>
</div>
<div id="vpravo">
        Pravý sloupec
        <p>I Zde bývají na různých portálech odkazy.</p>
</div>
</body>
</html>
```

Obr. 4.5 html kód s rozmístěním elementů `<div>` pro pozicování layoutu



Obr. 4.6 Výsledný layout webu podle zdrojového kódu z obr. 4.4 a 4.5

Pojem k zapamatování: Postup při tvorbě pozicování

Pro vytváření ručního CSS pozicování doporučuji nejprve návrh načrtnout na list papíru, poté zakreslit rozměry a nakonec vytvořit v CSS stylu názvy části stránek, ke kterým přiřadíme navržené rozměry.

4.4 Vytvoření CSS menu

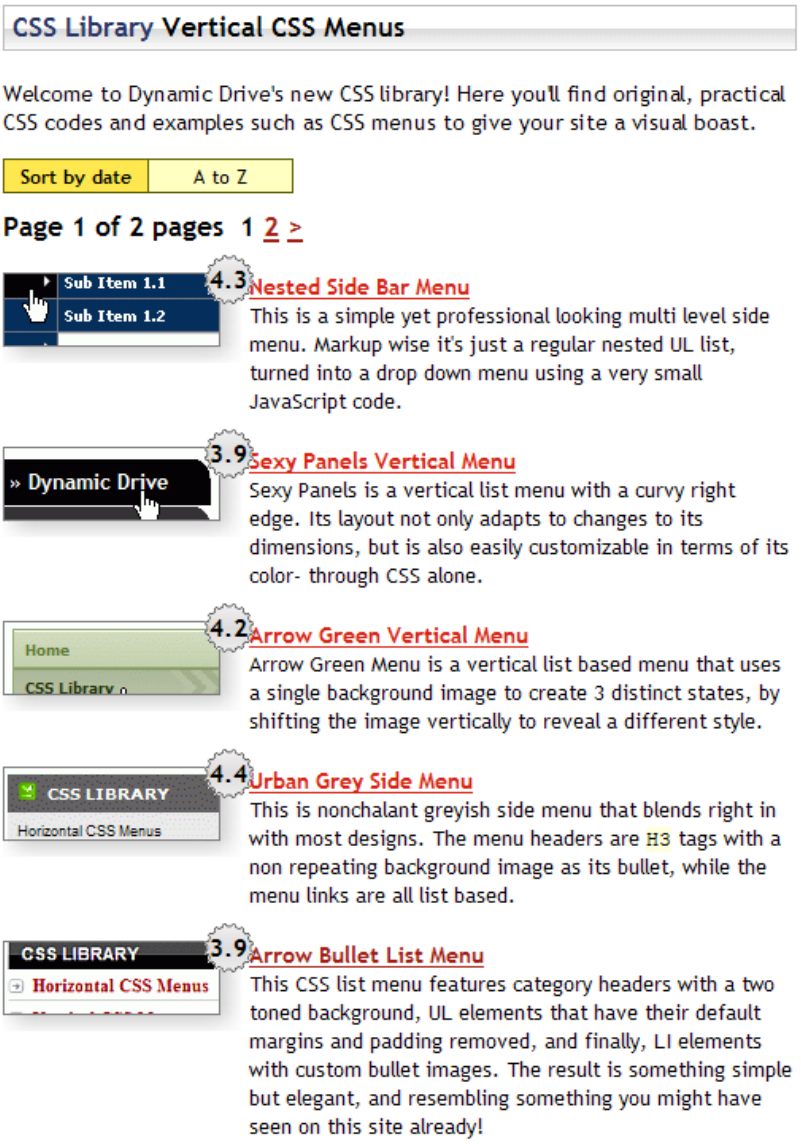
Pomocí CSS stylů máme možnost vytvářet profesionální návrh a layout stránek včetně pokročilých menu. Tyto menu můžeme buďto vytvářet sami, nebo máme možnost již vytvořená menu upravit pro naši potřebu. První varianta je velice pracná a vyžaduje nejen dlouholeté zkušenosti v oblasti návrhu stránek, ale bezesporu také talent v oblasti grafického cítění. Druhá varianta je snadnější a pro naše použití také doporučovanější.

Obr. 4.7 Vytvořené profesionální menu pomocí CSS stylů

Na obrázku 4.7 vidíme menu vytvořené pro naši potřebu. Menu je graficky zpracováno velice pěkně, pohybem myši po menu se barevně odlišuje aktuálně zvolená položka. Nyní si ukážeme celý postup, jak takové menu v krátké době vytvoříme.

Příklad 4.1 –Vytvoření navigačního menu v několika krocích

1. Vyhledejme vhodné menu pro naše webové stránky, to můžeme udělat ve vyhledávači volbou vhodných klíčových slov, např. *free css menu*. V našem příkladě budeme pokračovat na webu http://www.cssdrive.com.
2. Zvolíme volbu vertikální nebo horizontální menu, a vybereme pro naši potřebu vhodné menu, viz obr. 4.7.
3. Zvolené menu může obsahovat obrázky, proužky či jinou grafiku, kterou je nutné uložit do našeho projektu, viz. obr 4.8.
4. Zvolené menu obsahuje zdrojový kód CSS stylu, který importujeme do externího souboru, nebo přímo do zdrojového kódu naší stránky.
5. Pro rychlé otestování menu je nutné použít i zdrojový kód html viz. obr. 4.8 Pokud jej nepoužijeme, musíme definovat vlastní elementy `<div>` se selektorem id nadefinovaným v kaskádovém stylu.
6. Zdrojový kód menu upravíme tak, aby vyhovoval našim potřebám, tzn. upravíme názvy odkazů, jejich počet a cílové URL odkazů.



Obr. 4.7 Výběr profesionálního CSS menu

Tvorba profesionálních CSS menu

Pro vytvoření kvalitních menu pomocí CSS jej nemusíme složitě navrhovat. Kromě možností popsaných v této kapitole existují nástroje pro jejich generování.

- Webové nástroje:

1. http://www.cssmenumaker.com/
2. http://www.mycssmenu.com/
3. http://www.webmaster-toolkit.com/css-menu-generator.shtml

- Desktopové nástroje:

1. http://wonderwebware.com/css-menu/
2. http://cssmenu.com/

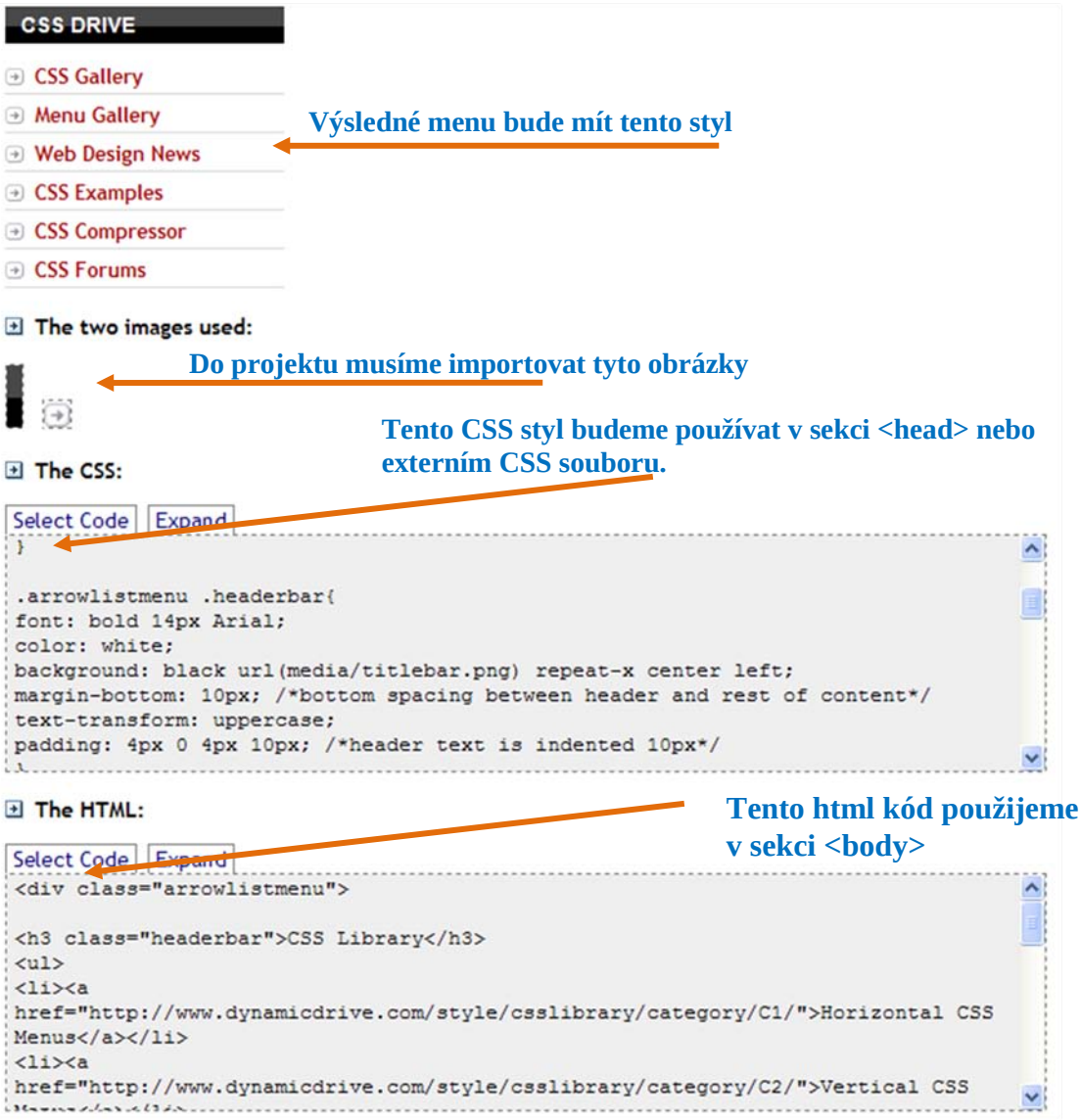
Tyto nástroje fungují na principu volby návrhu, barev, tvaru, dopsání názvu položek a následném vygenerování kódu stiskem tlačítka.

4.5 Používání dostupných šablon pro návrh vlastní aplikace

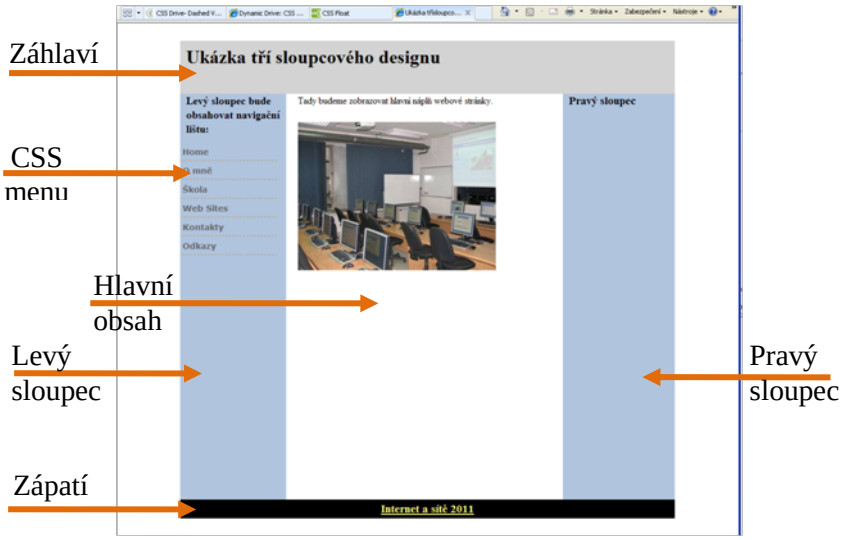
V kapitole 4.3 jsme se naučili vytvářet pozicovaný layout webové stránky. Nyní si obdobným způsobem jako menu z předchozí kapitoly vytvoříme layout pro naše stránky. Vytvoříme v několika rychlých krocích layout zobrazený na obr. 4.9.

Příklad 4.2 –Vytvoření navigačního menu v několika krocích

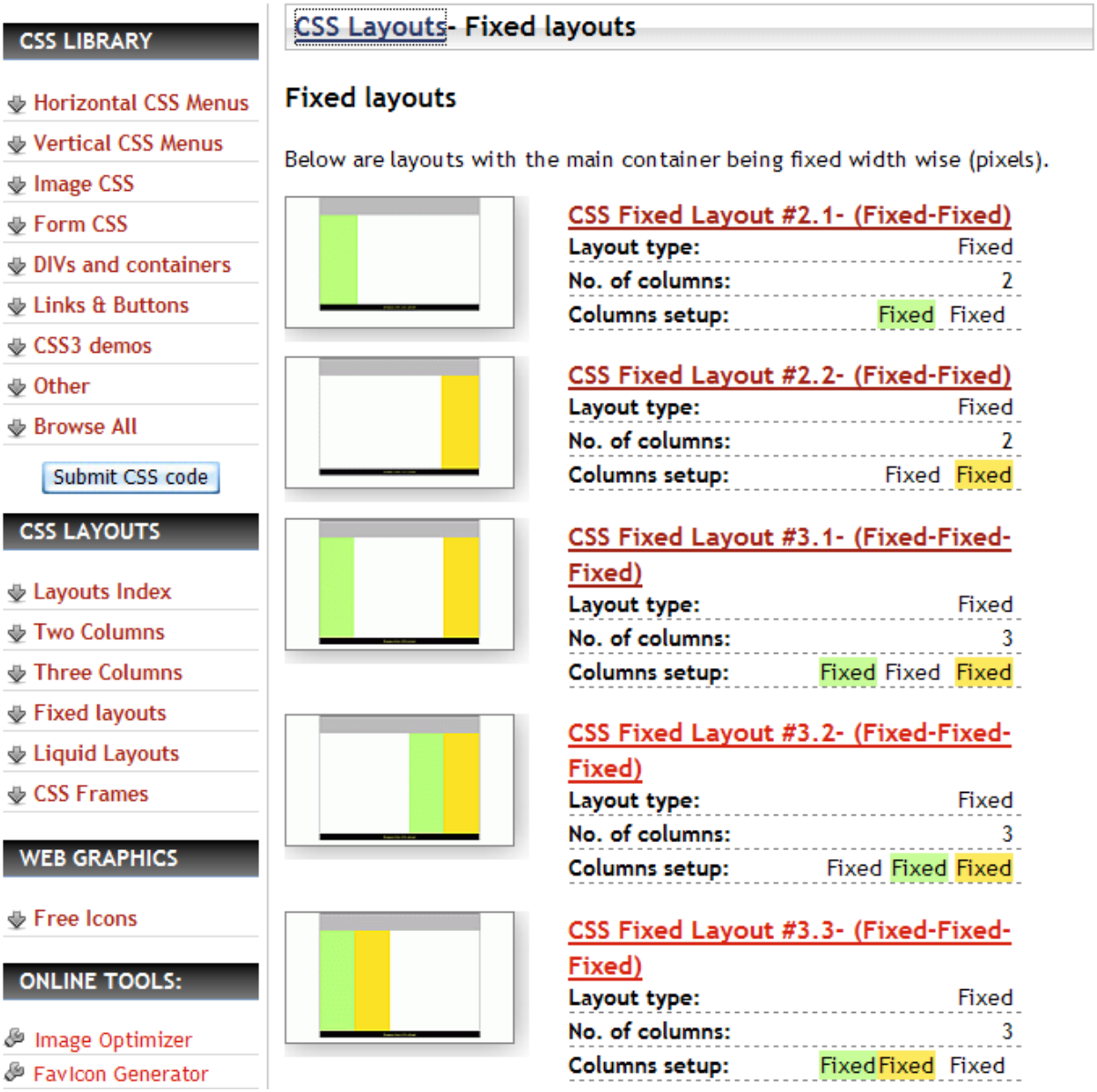
1. Vyhledejme vhodný layout pro naše webové stránky, opět to můžeme udělat ve vyhledávači volbou vhodných klíčových slov, např. *free css layout*. V našem příkladě opět použijeme server http://www.cssdrive.com.
2. Zvolíme volbu vhodného menu, v našem případě kategorii fixed layout, viz obr. 4.10.
3. Vybrané menu vidíme na obrázku s dostupným zdrojovým CSS kódem. Tento kód použijeme jako výchozí pozicování pro naši aplikaci.



Obr. 4.8 Použití selektoru ***vybraného menu pro vlastní aplikaci***



Obr. 4.9 Navržený web pomocí CSS layoutu



Obr. 4.10 Volba CSS layoutu z nabízených šablon

Obr. 4.11 Zvolený CSS layout a jeho zdrojový kód

Ke kapitole 4 je přiřazena demonstrační animace č. 2

Část animace č. 2 obsahuje praktickou ukázku tvorby layoutu web stránek pomocí CSS stylu. Rovněž animace předvádí vytvoření CSS menu s dostupných šablon.

Ke kapitole 4 je přiřazen demonstrační program č. 1

Program č. 1 demonstruje postup a ukázku kompletního zdrojového kódu pro vytvoření layoutu stránek pomocí CSS pozicování a vytvořené CSS template.

Shrnutí kapitoly

Layout webové stránky znamená její rozvržení na obrazovce a je nejdůležitější částí návrhu web stránky. Před samotným vytvořením layoutu musíme udělat analýzu, jaké prvky budou stránky obsahovat, kde a jaký bude informační obsah stránek. Podle toho také můžeme volit vícesloupcový layout, nebo layout se záhlavím, zápatím apod. Další prvky pak budeme umísťovat do předem určených míst, kde budou ve všech stránkách aplikace, tzn. např. navigační lišta s odkazy v levém sloupci, logo v hlavičce stránky, letopočet s copyrightem v zápatí, aktuality v pravém sloupci a podobně.

Rozložení obsahu stránek je řešeno různými způsoby, v různých dobách byly preferovány všechny. V současné době je doporučováno navrhovat rozmístění objektů na stránce pomocí CSS pozicování. Tabulkový layout či dokonce rámy jsou zastaralými technikami. Prohlížeče je sice podporují, ale přesto způsobují menší či větší problémy při návrhu či zobrazení takto zpracovaných stránek.

Pozicování elementů html nám umožňuje umístit objekt na stránku doslova, kam chceme. Pozicovat můžeme absolutně nebo relativně. Z názvu termínu *absolutně* vyplývá, že daný objekt se bude nacházet přesně v souřadnicích, které mu určíme. *Relativně* naopak znamená posunutí v různých směrech oproti běžné poloze. V tabulce 4.1 této kapitoly jsou definovány CSS vlastnosti pro pozicování elementů.

Pomocí CSS stylů máme možnost vytvářet profesionální návrh a layout stránek včetně pokročilých menu. Tyto menu můžeme buďto vytvářet sami, nebo máme možnost již vytvořená menu upravit pro naši potřebu. V této kapitole je popsán postup jak vytvářet kvalitní CSS menu podle dostupných online nástrojů. Stejně tak jako menu můžeme použít hotový layout stránek a upravit ho pro naši potřebu. Výsledná úprava spočívá v nastavení elementů `<div>` se selektorem id nadefinovaným v kaskádovém stylu. Zdrojový kód menu upravíme tak, aby vyhovoval našim potřebám, tzn., upravíme názvy odkazů, jejich počet a cílové URL odkazů.

Úkol k řešení 4.1 – Návrh layoutu stránek

Vytvořte ručně pozicovaný layout stránek, podle příkladu v kapitole 4.3.

Úkol k řešení 4.2 – Návrh CSS menu

Dle postupu výukového textu vytvořte pomocí online nebo desktopových nástrojů vertikální nebo horizontální menu, které použijete ve vašem layoutu stránek.

Úkol k řešení 4.3 – Generování layoutu online nástroji

Navrhněte vícesloupcový layout stránek. Pomocí dostupných nástrojů realizujte tento layout. Poté do místa, kde uvažujete navigační menu, umístěte menu vytvořené v předchozím úkolu. Vytvořte tak ucelenou kostru stránek s fungujícími odkazy mezi sebou.

Kontrolní otázka 4.1

Jaké jsou tři způsoby návrhu layoutu webových stránek?

Kontrolní otázka 4.2

Který způsob návrhu layoutu a proč preferujeme v současné době?

Kontrolní otázka 4.3

Jakým způsobem pozicujeme elementy html?

Kontrolní otázka 4.4

Které CSS vlastnosti používáme pro pozicování elementů?

Kontrolní otázka 4.5

K čemu slouží element `<div>`?

Kontrolní otázka 4.6

Jak se postupuje při tvorbě profesionálního CSS menu?

5. TVORBA WEBOVÝCH STRÁNEK POMOCÍ POKROČILÝCH NÁSTROJŮ

Čas ke studiu: 3 hodiny

Cíl Po prostudování této kapitoly budete umět

- používat profesionální nástroj pro tvorbu web stránek
- vytvořit webovou stránku v Microsoft Expression Web 3 studiu
- vyhledávat vhodné CSS šablony pro své html stránky
- editovat CSS šablony stránek v Microsoft Expression Web 3 studiu
- používat oficiální W3C CSS validátor
- odstraňovat chyby ve svých stránkách
- vytvořit kvalitní stránky pomocí profesionálního nástroje

Výklad

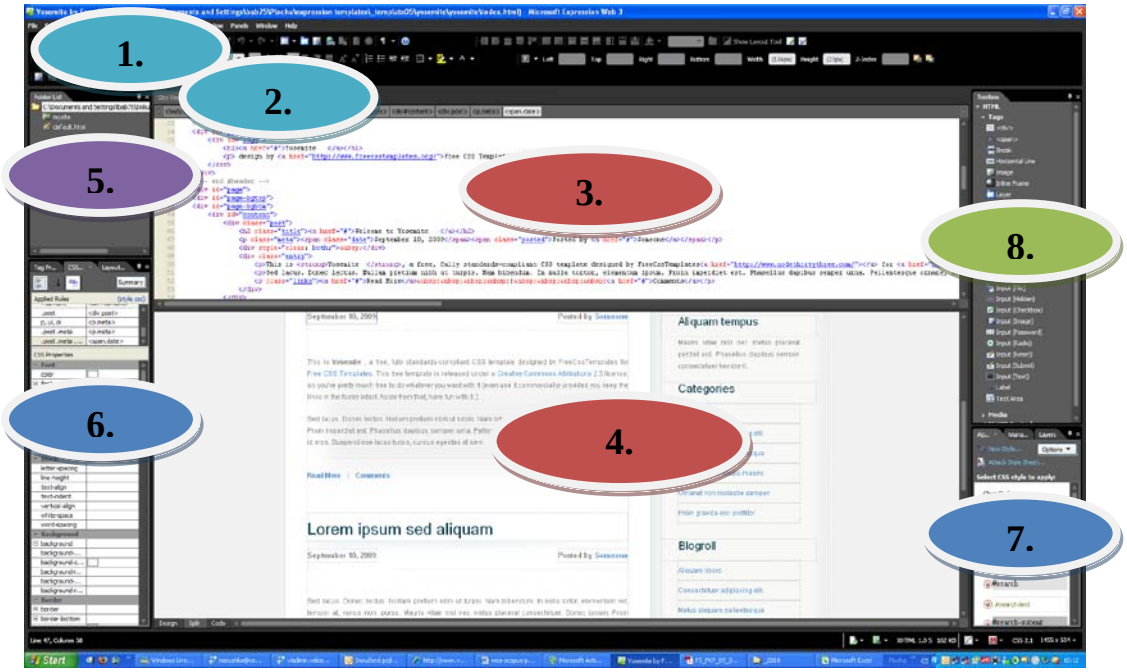
Během několika kapitol jsme pronikli do zákulisí tvorby webových stránek. Umíme stránky vytvářet, rozumíme html elementům a dokážeme stránky pozicovat a formátovat pomocí kaskádových stylů. Nyní si ukážeme práci v komfortním prostředí, které je pro tvorbu webových stránek určeno. V úvodní kapitole jsme popisovali seznam dostupných editorů, ve kterých můžeme stránky zdarma vytvářet, nyní si představíme profesionální nástroj, jehož licenci máme k dispozici v rámci akademické aliance. To znamená, že studenti všech forem studia mohou tento program využívat pro nekomerční použití.

5.1 Použití profesionálního editoru

V nedávné minulosti společnost Microsoft ve svém balíku kancelářských nástrojů vydávala různé verze programu FrontPage pro tvorbu a správu webových stránek. Postupem času a vývoje však tento editor nebyl konkurenceschopný profesionálnímu nástroji Dreamweaver společnosti Macromedia (dnes již Adobe). Microsoft si tohoto faktu byl vědom, a tak vyvinul nástroj Microsoft Web Expression. V současné době je tento plně profesionální nástroj ve verzi 3, je určen profesionálním designérům, ale také běžným uživatelům, kteří chtějí využít pro svou práci vynikajícího prostředí.

Microsoft Expression Web je založen na standardech dnes moderních technologií jako XHML, XML, XSLT a CSS a jeho vykreslovací jádro přesně interpretuje kód založený na validaci XHTML. Mimo základních XHTML stránek je možné tvořit webové aplikace s podporou serverových skriptovacích jazyků ASP.NET či PHP. Microsoft Expression Web umožňuje tvořit webové stránky komfortním způsobem pomocí vestavěných prvků, spravovat návrhy a styly pomocí myši, využívat předdefinované šablony, spravovat přehledně vlastnosti elementů a v neposlední řadě využívat nástroj *intellisense*, který umožňuje přímo při psaní zdrojového kódu nabízet vývojářům zdrojový kód, možné vlastnosti elementů a CSS, které během psaní kódu postupně zpřesňuje a doplňuje nabízené možnosti dle standardů a mimo jiné ukončuje párové elementy. Vývojář tak má jistotu, že nepoužije nevhodný či nestandardizovaný zdrojový kód, jeho práce je navíc snadnější a rychlejší. Novou vlastností tohoto produktu je možnost prohlížení výsledné stránky ve více prohlížečích libovolných verzí vedle sebe, což jistě ocení mnozí vývojáři při dnešní existenci mnoha prohlížečů. Na obrázku 5.1 vidíme pracovní plochu prostředí, kterou si nyní popíšeme. V horní části máme možnost nastavovat lišty pracovních

nástrojů, tak jak nám nejlépe vyhovují a podle toho, které budeme nejčastěji používat – (část 1). Pod pracovními lištami se nachází záložky právě otevřených souborů (část 2). Tyto záložky umožňují editaci mnoha souborů projektů najednou, bez nutnosti znovu otevírání a ukládání souborů. Největší plochu zabírá zobrazení zdrojového kódu a grafického návrhu stránek. Zdrojový kód či vzhled stránky můžeme zobrazovat samostatně každé zvlášť do jednoho velkého prostoru, nebo současně pod sebou. K tomuto zobrazení slouží přepínač v levém dolním rohu, který je zobrazen na obr. 5.2. V případě zobrazení pod sebou si můžeme všimnout, že při editaci či vývoji návrhu stránky (část 4) se přímo generuje zdrojový kód (část 3).

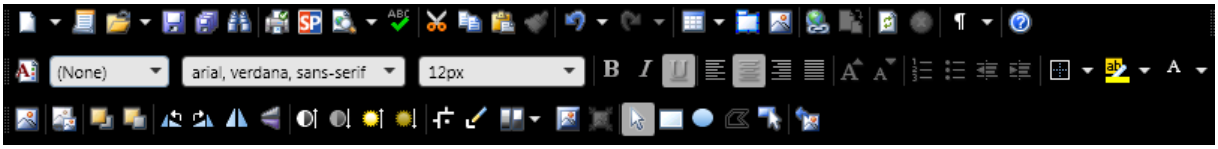


Obr. 5.1 Rozvržení pracovní plochy nástroje Microsoft Expression Web

Celou pracovní plochu si můžeme přizpůsobit vlastním požadavkům. Celou řadu panelů můžeme umístit do vyhovujících části plochy, nejčastěji však na levou a pravou stranu. Mezi nejčastěji používané panely jsou: Průzkumník složek – (část 5), okno vlastností elementů (část 6) a CSS vlastností (část 7) a panel pracovních nástrojů, tzv. *toolbox* (část 8).

Obr 5.2 Přepínač zobrazení návrhu stránek a zdrojového kódu

Pracovní plochu prostředí si můžeme přizpůsobit pomocí menu *View* → *Toolbars*. Zde můžeme vidět více než deset kategorií nástrojů, které se pomocí ikon zobrazí na nástrojové liště, viz. obr. 5.3. Můžeme tak nejpoužívanější nástroje mít ihned k dispozici a naopak, ty které nepoužíváme, můžeme na nějaký čas skrýt.

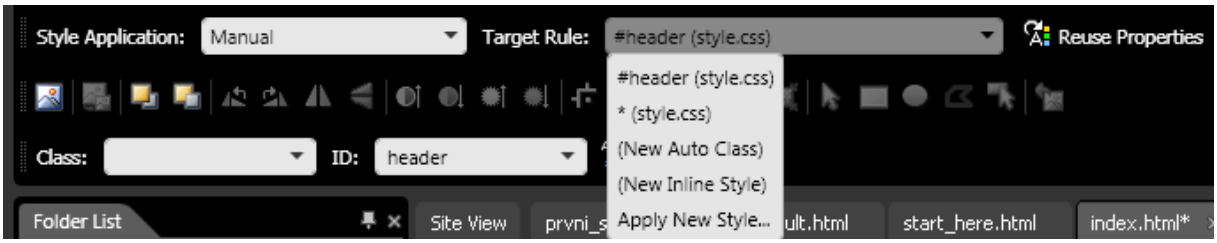


Obr 5.3 Příklady rozložení pracovní plochy Expression Web 3 studia

Moderní způsob tvorby webu prostřednictvím kaskádových stylů má za cíl oddělit návrh vzhledu od obsahu. Expression Web zajišťuje tuto složitou technologii svými nástroji určenými pro design. Uživatel má plnou kontrolu nad tím, jak bude stránka vypadat, od nastavování okrajů a výplní pomocí myši po vizuální použití stylů. Na obrázku 5.4 vidíme lištu pozicování, ve které můžeme ihned navrhovat nebo editovat pozicování jednotlivých elementů.

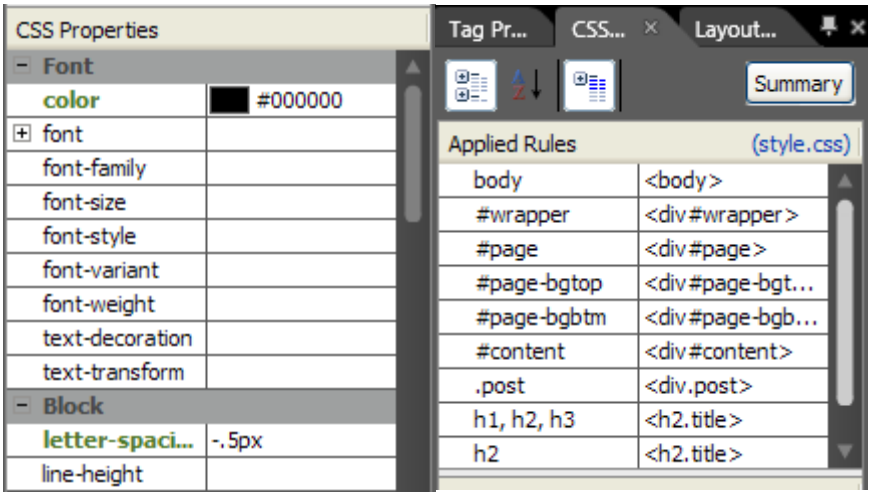
Obr 5.4 Nástroje pro nastavení souřadnic CSS pozicování

Expression Web umožňuje volbu mezi automatickým generováním CSS nebo jejich ručním ovládáním, které dovoluje nastavení vytvářeného pravidla a jeho přesné umístění. Na obr. 5.5 vidíme volbu vlastního pravidla pro nastavení CSS stylu, konkrétně pro element `<div id="header">`.



Obr 5.5 Nástroje pro nastavení souřadnic CSS pozicování

Podokna vlastností zajišťují rychlý přístup ke kompletní škále vlastností CSS a značek. Seznam si můžete uspořádat tak, abyste snadno našli ty, které potřebujete, a aby se zobrazily jen ty vlastnosti CSS, které jsou právě použity. Můžeme v záložkovém menu přepínat mezi hodnotami *Tag properties* nebo *CSS properties*.



Obr 5.6 Nástroje pro nastavení souřadnic CSS pozicování

5.2 Pokročilé webové stránky pomocí CSS šablon

Pro návrh layoutu vlastní webové prezentace můžeme zvolit dva typy řešení. Prvním složitějším řešením je tvořit celý návrh svými prostředky od začátku. Návrh stránky je tak stavěn doslova na zelené louce. Tento způsob je pracný, avšak můžeme své představy o vzhledu realizovat pomocí pozicovaných elementů, které navrhneme na míru našim požadavkům. Tento postup jsme si vyzkoušeli v minulé kapitole. Druhou možností je náš návrh přizpůsobit již hotové profesionální šabloně popřípadě pokud nechceme slevit z našich požadavků, nejvhodnější šablonu dále editovat. Tento způsob tvorby pokročilých webových stránek má mnoho výhod. Kromě daleko menší časové náročnosti je to určitě propracovaný grafický design, kterého bychom bez profesionálních grafických nástrojů a hlavně bez perfektní tvůrčí dovednosti a grafického cítění těžko dosáhli. Jedinou nevýhodou je nadbytečnost některých prvků na stránce, kterých se ale při editaci šablony jednoduše zbavíme. Odměnou nám bude profesionální propracovaný layout s možností efektů přepínání v menu, různých záložkových menu se změnou podsvícení a barev, které bychom museli pracně obsluhovat javascriptem či pokročilým CSS formátováním. Postup je tedy následující:

1. Vytvoříme požadavky na layout našeho webového sídla.

2. Na serverech věnovaných CSS šablonám vyhledáme nejvhodnější CSS *template*.

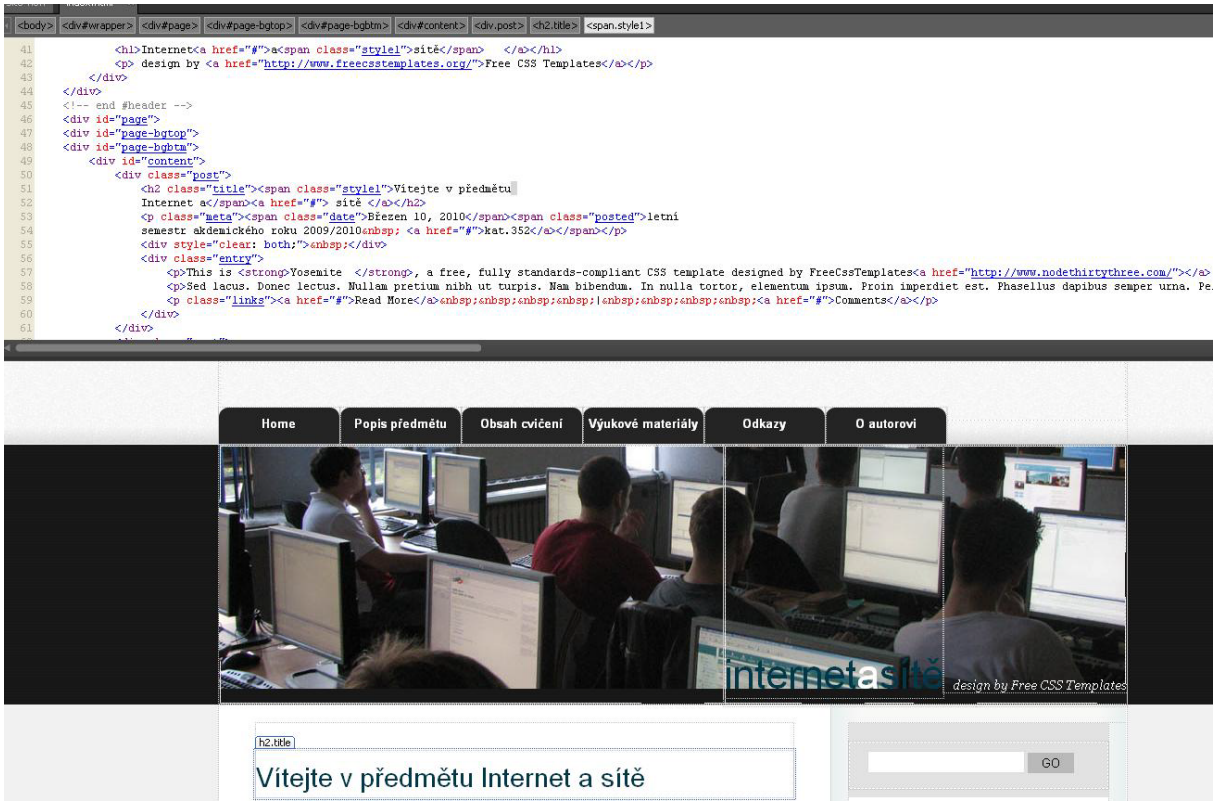
Seznam serverů s profesionálně propracovanými šablonami zdarma

1. http://www.freecsstemplates.org/
2. http://www.free-css.com/free-css-templates/page1.php
3. http://www.free-css-templates.com/
4. http://www.free-template.eu/
5. http://www.freecsstemplates.com/
6. http://www.freecsstemplate.net/free-css-templates.php

A mnoho dalších, stačí použít vyhledávač. Na každém serveru jsou desítky až stovky šablon, které jsou připravené pro vaši editaci.

3. CSS šablonu upravíme pomocí našich požadavků. Upravíme počet položek menu, změníme linky, odstraníme nepotřebné prvky webu, popřípadě upravíme na míru rozměry.
4. Doplníme vlastní grafické prvky, obrázky a naplníme informačním obsahem všechny stránky webového sídla.

Na obrázku 5.7 vidíme editovanou CSS šablonu. Nejprve bylo upraveno záložkové menu a odkazy, poté upraven hlavní obrázek *homepage* dle předepsaných rozměrů šablony. Následovat bude plně informačního obsahu *homepage* a dále pak zbylých pěti stránek aplikace.



Obr 5.7 Tvorba webu k předmětu pomocí CSS template

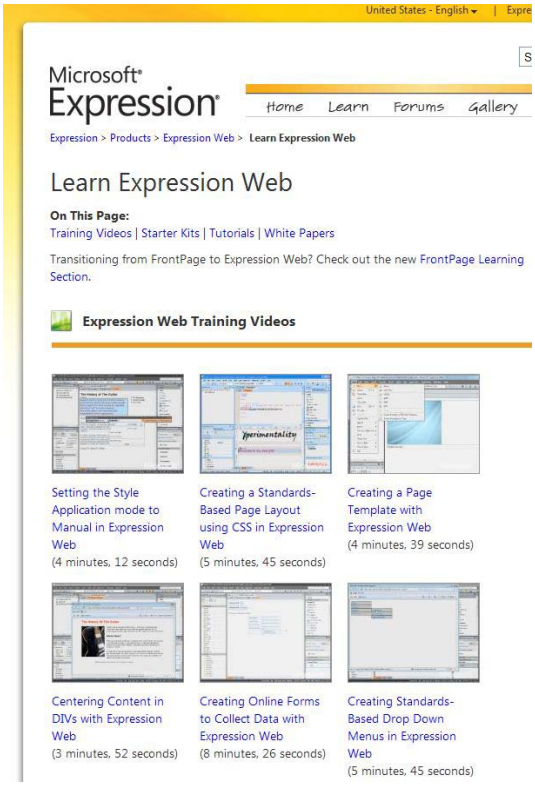
5.3 Výukový server Expression Web 3 studia

Nejlepším prostředkem pro pochopení práce ve vývojovém prostředí je přímo procvičení praktických dovedností na příkladech. Microsoft Expression Web 3 studio je mocný nástroj a popis jeho možností by zabral stovky stran. Proto společnost Microsoft uvolnila sadu webcastů zdarma pro pochopení práce v tomto vývojovém prostředí. Není tedy nutné kupovat drahé knihy, je třeba se ale připravit na anglický popis prezentací. V následujícím seznamu vidíme krátký výčet doporučených prezentací pro zvládnutí práce ve vývojovém prostředí Microsoft Expression Web 3.

Seznam dostupných návodů pro tvorbu web stránek v prostředí Microsoft Expression Web 3 ve formě webcastů

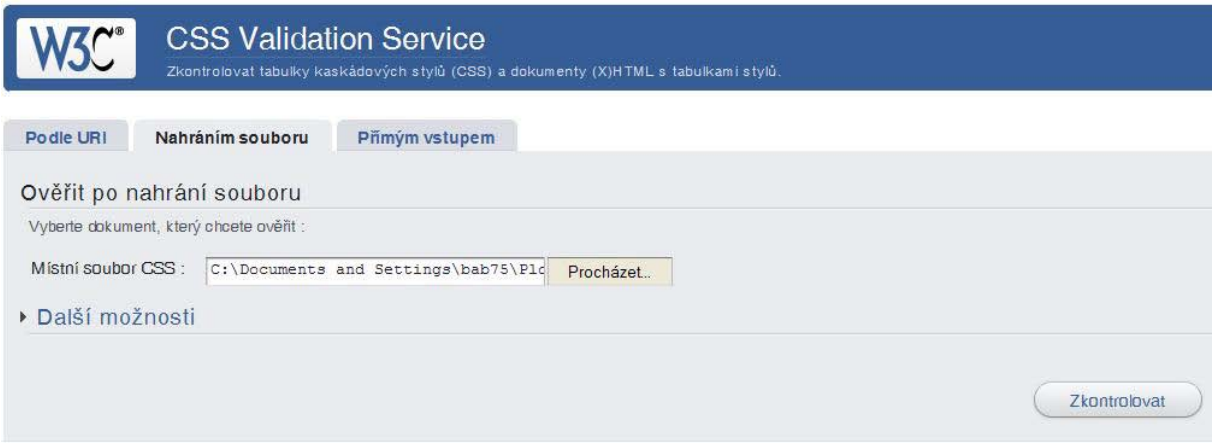
http://expression.microsoft.com/ je server společnosti Microsoft na kterém najdeme mnoho výukových webcastů, které nám předvedou profesionální práci s tímto prostředím při tvorbě web stránek.
Mimo jiné zde najdeme:

- Základní použití prostředí
- Tvorba Master Page
- Vytvoření šablony stránek
- Tvorba navigačního menu
- Použití sofistikovaných CSS šablon
- Práce s odkazy
- Tvorba DropDownList menu
- Tvorba portálového webu
- Vytvoření formulářů
- Pokročilá práce s XML dokumenty
- Použití pokročilých technologií jako AJAX, Silverlight, ASP.NET a další.

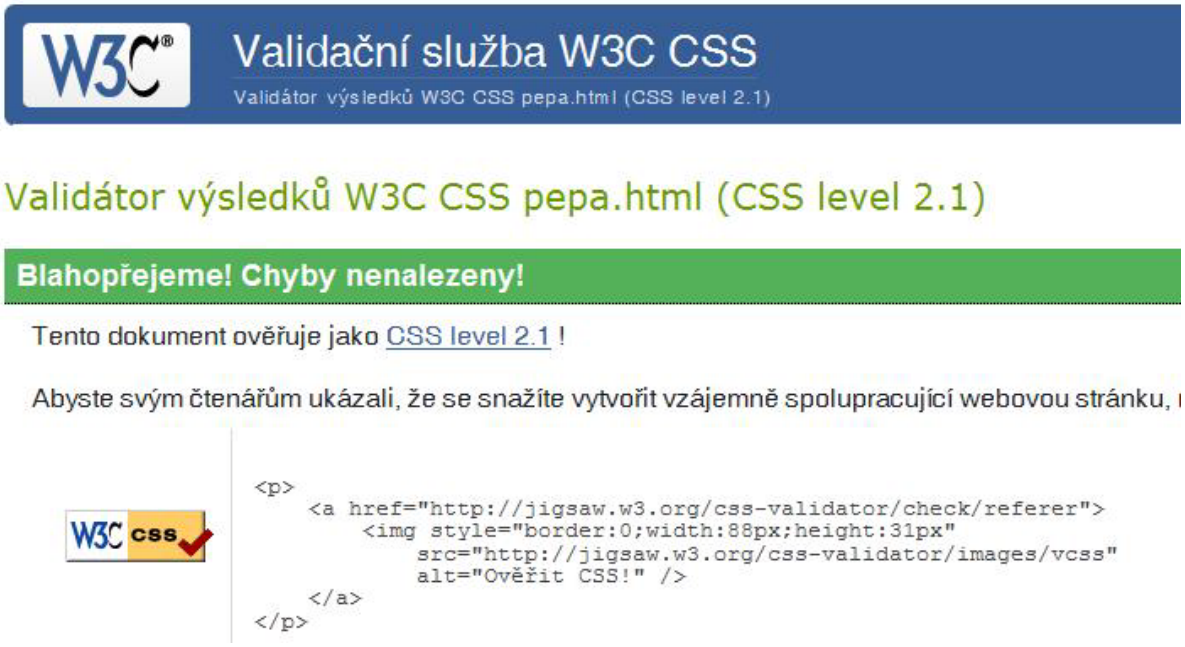


5.4 CSS a XHTML validace

W3C konsorcium nabízí online nástroje pro validaci vašich vytvořených web stránek. Jsou dostupné z více míst, například CSS validátor je dostupný z URL adresy http://jigsaw.w3.org/css-validator/, xhtml validátory z adresy http://validator.w3.org/.
Pomocí těchto služeb si můžeme ověřit, zda-li naše stránky odpovídají standardům CSS a XHTML. To můžeme provést uploadováním stránky se zdrojovým kódem – viz. obr. 5.8, vložením zdrojového kódu přímo do formuláře nebo zadáním URI adresy, pokud už je stránka na webovém serveru. V případě úspěšné validace uvidíme výstup se zelenou lištou a oznámením, že žádné chyby nebyly nalezeny viz. obr. 5.9, v opačném případě bude lišta červená s hlášením o počtu chyb. V tomto případě uvidíme výpis jednotlivých chyb a varování s číslem řádku zdrojového kódu.



Obr. 5.8 CSS validátor webové stránky



Obr. 5.9 Zobrazení úspěšného výsledku validace

Ke kapitole 5 je přiřazena demonstrační animace č.3

Část animace č. 3 obsahuje praktickou ukázku práce s úpravou CSS šablony web stránky v prostředí Microsoft Expression Web 3.

Ke kapitole 5 je přiřazen demonstrační program č.2

Program č. 2 demonstruje postup a ukázku kompletního zdrojového kódu pro vytvoření webových stránek pomocí pokročilé CSS template.

Shrnutí kapitoly

Microsoft Expression Web je založen na standardech dnes moderních technologií jako XHML, XML, XSLT a CSS a jeho vykreslovací jádro přesně interpretuje kód založený na validaci XHTML. Mimo základních XHTML stránek je možné tvořit webové aplikace s podporou serverových skriptovacích jazyků ASP.NET či PHP. Microsoft Expression Web umožňuje tvořit webové stránky komfortním způsobem pomocí vestavěných prvků, spravovat návrhy a styly pomocí myši, využívat předdefinované šablony, spravovat přehledně vlastnosti elementů a v neposlední řadě využívat nástroj *intellisense*, který umožňuje přímo při psaní zdrojového kódu nabízet vývojářům zdrojový kód, možné vlastnosti elementů a CSS, které během psaní kódu postupně zpřesňuje a doplňuje nabízené možnosti dle standardů a mimo jiné ukončuje párové elementy. Vývojář tak má jistotu, že nepoužije nevhodný či nestandardizovaný zdrojový kód, jeho práce je navíc snadnější a rychlejší.

Pracovní plochu prostředí Microsoft Expression Web 3 si můžeme přizpůsobit pomocí menu *View* → *Toolbars*. Zde můžeme vidět více než deset kategorií nástrojů, které se pomocí ikon zobrazí na nástrojové liště. Moderní způsob tvorby webu prostřednictvím kaskádových stylů má za cíl oddělit návrh vzhledu od obsahu. Expression Web zajišťuje tuto složitou technologii svými nástroji určenými pro design. Uživatel má plnou kontrolu nad tím, jak bude stránka vypadat, od nastavování okrajů a výplní pomocí myši po vizuální použití stylů. Expression Web umožňuje volbu mezi automatickým generováním CSS nebo jejich ručním ovládáním, které dovoluje nastavení vytvářeného pravidla a jeho přesné umístění. Nejlepším prostředkem pro pochopení práce ve vývojovém prostředí je přímo procvičení praktických dovedností na příkladech. Microsoft Expression Web 3 studio je mocný nástroj a popis jeho možností by zabral stovky stran. Proto společnost Microsoft uvolnila sadu webcastů zdarma pro pochopení práce v tomto vývojovém prostředí: http://expression.microsoft.com/.

Pro návrh layoutu vlastní webové prezentace můžeme zvolit dva typy řešení. Prvním složitějším řešením je tvořit celý návrh svými prostředky od začátku. Druhou možností je náš návrh přizpůsobit již hotové profesionální šabloně popřípadě pokud nechceme slevit z našich požadavků, nejvhodnější šablonu dále editovat. Postup je tedy následující: Vytvoříme požadavky na layout našeho webového sídla. Poté na serverech věnovaných CSS šablonám vyhledáme nejvhodnější CSS *template*. CSS šablonu upravíme pomocí našich požadavků. Upravíme počet položek menu, změníme linky, odstraníme nepotřebné prvky webu popřípadě upravíme na míru rozměry. Doplníme vlastní grafické prvky, obrázky a naplníme informačním obsahem všechny stránky webového sídla.

W3C konsorcium nabízí online nástroje pro validaci vašich vytvořených web stránek. Jsou dostupné z více míst, například CSS validátor je dostupný z URL adresy http://jigsaw.w3.org/css-validator/, xhtml validátory z adresy http://validator.w3.org/. Pomocí těchto služeb si můžeme ověřit, zdali naše stránky odpovídají standardům CSS a XHTML.

Úkol k řešení 5.1 – Vyhledání vhodné CSS šablony

Prohlédněte si odkazy na servery s dostupnými volně stažitelnými CSS šablonami. Na základě vašich požadavků vhodnou šablonu stáhněte a otevřete v Microsoft Expession Web 3 studiu.

Úkol k řešení 5.2 – Vytvoření vlastní webové stránky

Šablonu z předchozího úkolu editujte a vytvořte svůj vlastní web.

Úkol k řešení 5.3 – práce s W3C validátorem

Vyzkoušejte validaci pomocí W3C validátory všemi možnostmi: zadáním URL nějaké stránky, nahráním stránky pomocí *upload* formuláře a vložením části kódu pro kontrolu.

Úkol k řešení 5.4 – Validace vytvořené stránky

Vámi vytvořenou stránku validujte pomocí W3C validátoru.

Kontrolní otázka 5.1

Co je to IntelliSense?

Kontrolní otázka 5.2

Jaké typy validace W3C máme k dispozici?

Kontrolní otázka 5.3

Jakým způsobem validujeme kaskádové styly podle specifikaceW3C?

Kontrolní otázka 5.4

Jakým způsobem v pokročilých vývojových prostředích zobrazujeme současně návrh stránek a jejich zdrojový kód?

6. POUŽITÍ JAVASCRIPTU

Čas ke studiu: 3 hodiny

Cíl Po prostudování této kapitoly budete umět

- vyjmenovat důvody pro použití javascriptu
- popsat možnosti a schopnosti skriptovacího jazyka
- používat vybrané objekty javascriptu
- vhodně vytvářet a zapisovat javascriptový kód do html stránek
- vyhledat a použít skripty na míru vaší aplikaci pro požadované dynamické chování webových stránek

Výklad

Nyní již máme za sebou výklad o HTML jazyku, kaskádových stylech, pozicování stránek a umíme vytvářet stránky na poměrně kvalitní úrovni. Stále však na stránkách můžeme postrádat určitou dynamiku. Komunikace klienta s webovým serverem neumožňuje pomocí html kódu zajistit dynamické chování v klientském počítači. Nemůžeme tak kontrolovat obsahy formulářů, *cookies*, měnit dynamicky text a odpovídat na určité události na straně klienta, protože o tom web server nic neví. Řešením této situace jsou skriptovací jazyky. Javascript se skriptovací jazyk na straně klienta, který je určen jako nástroj pro výše uvedené skutečnosti. Tato kapitola není referenční programátorskou příručkou pro javascript, vysvětlí nám však jeho principy a naučí javascripty v plné míře využívat.

6.1 Úvod do použití Javascriptu v HTML stránkách

K čemu vůbec potřebujeme javascript? Než pronikneme do konkrétních vlastností a syntaxe jazyka, řekněme si co vlastně umí v interakci s jazykem HTML:

- Dokáže reagovat na události jako klikání myší, změna polohy objektů apod.
- Pomocí něj můžeme validovat data z formulářů na straně klienta.
- Dokáže vytvořit dynamický text v HTML stránce.
- Dokáže přizpůsobovat a měnit HTML elementy.
- Umožňuje číst a zapisovat proměnné *cookies*, detekovat klientův prohlížeč a další užitečné vlastnosti.

Pojem k zapamatování: Skript na straně klienta

Javascript je skriptovací jazyk na straně klienta, to znamená, že webový server nemá žádnou informaci o tom co se na klientském počítači děje. Všechny akce, které provádí skript se dějí na klientské straně.

Oproti tomu, technologie jako ASP.NET používá pro komunikaci s klientem serverové ovládací prvky a ty pak již zpracovávají data na webovém serveru a zasílají klientovi kód html na míru.

Site View | prvni_script.html*

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www
2  <html xmlns="http://www.w3.org/1999/xhtml">
3
4  <head>
5  <meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
6  <title>Použití javascriptu </title>
7  <script type="text/javascript">
8  function vzkaz()
9  {
10 alert("Tento skript použijeme při načtení stránky, při události onload");
11 }
12 </script>
13 </head>
14 <body onload="vzkaz()">
15 <script type="text/javascript">
16 document.write("Skript můžeme nadefinovat i použít v těle dokumentu");
17 </script>
18 </body>
19 </html>
```

Obr. 6.1 Umístění javascriptového kódu do sekce <head> nebo <html>

6.2 Proměnné a operátory

Každý jazyk, ať už programovací či skriptovací pracuje s proměnnými. Proměnné se obvykle deklarují a jsou určitého datového typu. Slovo obvykle je tu záměrně protože právě pro javascript tato charakteristika neplatí. Proměnné můžeme, ale nemusíme deklarovat a datový typ k nim nepřiřazujeme, potřebujeme pouze vědět, zda-li se jedná o řetězec, logickou hodnotu či číslo a o tom rozhodnou uvozovky. Proměnná může obsahovat ještě logickou hodnotu *true* nebo *false* (pravda/nepravda). Pokud proměnnou deklarujeme, používáme klíčové slovo *var*.

Příklad 6.1 deklarace proměnných

```
<script type="text/javascript">
   var vek;
   var jmeno = "Karel";
   var b = 5;

   vek = b;

   document.write(jmeno + " má " + vek + " let")
</script>
```

V příkladu vidíme deklaraci proměnných, proměnným můžeme při deklaraci přiřadit hodnotu. Proměnné *vek* jsme nepřiřadili hodnotu, ostatním proměnným ano. Proměnná *jmeno* je textová, proto je její hodnota v uvozovkách. Ve skriptu jsme dále vypsali hodnoty proměnných. Proměnné jsme spojili operátorem + do řetězce a připojili i několik řetězců, které musí být na rozdíl od proměnných v uvozovkách.

Pojem k zapamatování: Javascript je Case-sensitive jazyk

Javascript je case sensitive, to znamená, že na rozdíl od HTML rozlišuje velká a malá písmena. Proměnná *Jmeno* je tedy odlišná od proměnné *jmeno*.

Již v prvním příkladu jsme použili operátor +. Tento operátor se používá jako aritmetický operátor, ale také jako operátor spojování řetězců. Mezi Aritmetické operátory dále patří nejčastější operace (odčítání, násobení, dělení a modulo s operátory -,* / a %). Dalšími operátory je inkrementace a dekrementace neboli zvětšení a zmenšení hodnoty proměnné o 1. K těmto operacím jsou určeny operátory ++ a --.

Pojem k zapamatování: rozdílná syntaxe komentářů

Pokud budeme chtít okomentovat náš kód nebo vložit poznámku můžeme to v javascriptu udělat následujícím způsobem:

// komentář na jeden řádek je za dvěma lomítky

/ komentář na více*
*řádků je uvozen lomítkem a hvězdičkou */*

Nezapomeňme, že pro jazyl html platí zcela jiné pravidlo a komentář je umístěn mezi značky elementu, který používá vykřičník a dvě pomlčky:

<!-- komentář v html -->

6.3 Syntaxe příkazů Javascriptu

Javascript je jazyk vycházející ze syntaxe Javy či jazyka C. Kromě metod objektů, které si předvedeme na příkladech v následující kapitole, budeme často využívat určité příkazové konstrukce. Ke každému jazyku programovacímu či skriptovacímu patří programové konstrukce jako porovnávání, rozhodování, větvení, cykly a funkce. V této kapitole uvedeme syntaxe jednotlivých konstrukcí a v připraveném programu si je na demonstračních příkladech procvičíme.

Tab. 6.1 Syntaxe a význam podmínek

Podmínkové rozhodování		
if	if (podmínka) {sekvence příkazů}	Pokud je splněna podmínka bude provedena sekvence příkazů
if else	if (podmínka) {sekvence 1} else {sekvence 2}	Pokud je splněna podmínka bude provedena sekvence příkazů 1, pokud podmínka splněna není, bude provedena sekvence příkazů 2
if else if	if (podmínka 1) {sekvence 1} else if (podmínka 2) {sekvence 2} else {sekvence 3}	Pokud je splněna podmínka 1 bude provedena sekvence příkazů 1, pokud podmínka splněna není, bude provedeno rozhodování o další podmínce, pokud ta bude splněna, bude provedena sekvence příkazů 2, pokud ne, bude provedena sekvence příkazů 3.

Sekvence příkazů provedena v podmínkovém rozhodování musí být v závorkách, za každým příkazem musí následovat středník. U složitějších rozhodování můžeme mít vícenásobný počet if –*else if*. Zvláštnosti jazyků odvozených od jazyka C je rozhodovací operátor **?**, který umožní rychlejší zápis rozhodovací podmínky, pokud má být sekvencí příkazů pouze přiřazení proměnné. Jeho tvar je následující:

proměnná = podmínka ? hodnota1: hodnota2

Pokud je podmínka splněna do proměnné se uloží hodnota 1, pokud proměnná splněna není, do proměnné se uloží hodnota 2.

Tab. 6.2 Syntaxe a význam větvení

Větvení		

switch	switch (proměnná) {
	case hodnota: příkaz1;break;
	case hodnota2:příkaz2;break;
	case hodnota n:příkaz n;break;
	default:příkaz x;
	}
		Při větvení do více alternativ je provedena pouze větev při splněné podmínce, pokud není žádná podmínka slněna je provedena varianta v návěští default. Za každou větví musí následovat příkaz break, jinak by bylo provedeno více větví programu.

Následuje přehled syntaxe cyklů, ty bývají v programovacích jazycích s podmínkou na začátku, s podmínkou na konci a s přesným počtem opakování. Pokud máme podmínku na konci, je jasné, že tělo cyklu bude provedeno alespoň jednou.

Tab. 6.3 Syntaxe a význam cyklů

Cykly		
while	while (podmínka) {sekvence příkazů}	Dokud je splněna podmínka bude prováděna sekvence příkazů
do while	do {sekvence příkazů} while (podmínka)	Dokud je splněna podmínka bude prováděna sekvence příkazů
for	for (počátek;podmínka;navýšení) {sekvence příkazů}	Cyklus bude prováděn od počáteční hodnoty, dokud bude splněna podmínka, porovnávaná proměnná bude navyšována

Rovněž v cyklech můžeme použít příkaz *break* pro předčasné ukončení provádění těla cyklu.

Ke kapitole 6 je přiřazen demonstrační program č.3

Program č. 3 demonstruje postup a ukázku kompletních zdrojových kódů pro pochopení syntaxe příkazových konstrukcí javascriptu.

6.4 Objektový model a vybrané objekty

Javascript je částečně objektový jazyk, nevyužívá možnosti objektově orientovaného programování jako pokročilé OOP jazyky. U javascriptu se pod pojmem objektový chápe jisté uspořádání systému a přístup k objektům pomocí metod a vlastností. K objektům a podobjektům se přistupuje tečkovou konvencí rovněž metoda objektu je volána způsobem objekt.metoda(). Přístup k uspořádání objektů na stránce nám napoví následující příklad převodu teplot z Celsia na Fahrenheita pomocí javascriptu. Představme si že budeme mít v dokumentu formulář obsahující textbox s hodnotou proměnné. Pak se podle tečkové konvence na tuto proměnnou budu odkazovat:

dokument.formulář.textbox.proměnná.její_hodnota

Pro převod teploty vytvoříme dva textboxy, kde budeme zobrazovat teplotu ve stupních celsia a fahrenheita. Nebudeme vytvářet žádná tlačítka, převod provedeme na základě události ***onChange***, což znamená, že pokud se změní obsah textboxu, dojde k vyvolání události. Stačí nám tedy kdekoliv na stránce kliknout tlačítkem myši.

Pro převod mezi soustavami využijeme vzorec F= 9/5 C + 32. K tomuto výpočtu nám bude stačit použití základních aritmetických operátorů, pokud bychom však chtěli využít složitější matematickou konstrukci jako odmocnina, mocnina, zaokrouhlení apod., museli bychom využít metody objektu *Math*, které uvádíme v tabulce 6.1.

Tab. 6.4 Metody objektu Math

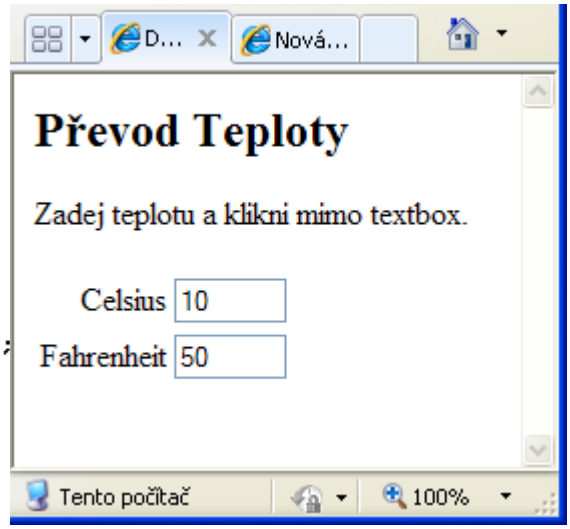
Metoda	Popis	Metoda	Popis
abs(x)	absolutní hodnota	**asin(x)**	arcus sinus
exp(x)	e na x-tou	**atan(x)**	arcus tangens
log(x)	přirozený logaritmus	**Atan2(x,y)**	úhel v rad od osy x do bodu x,y
max(x,y)	větší ze dvou hodnot	**cos(x)**	cosinus
min(x,y)	menší ze dvou hodnot	**sin(x)**	sinus
pow(a,x)	a na x-tou	**tan(x)**	tangens
random()	náhodné číslo mezi 0 a 1	**ceil(x)**	zaokrouhlení nahoru
Sqrt(x)	odmocnina	**floor(x)**	zaokrouhlení dolů
acos(x)	arcus cosinus	**round(x)**	zaokrouhlení aritmetické

V našem příkladu tedy využijeme metodu *Math.round* pro aritmetické zaokrouhlení převedené teploty.

```
17 <body>
18 <h2>Převod Teploty</h2>
19 <p>Zadej teplotu a klikni mimo textbox.</p>
20 <form name="f">
21 <table>
22 <tr><td align="right">Celsius </td>
23 <td><input type="text" name="celsius" size="5" onChange="C2F()"></td>
24 <tr><td align="right">Fahrenheit</td>
25 <td><input type="text" name="fahrenheit" size="5" onChange="F2C()"></td>
26 </table>
27 </form></body></html>
```

Obr. 6.2 html kód pro příklad převodu teploty

```
1  <html>
2  <head>
3  <script type="text/javascript">
4   function C2F() {
5    var C = eval(document.f.celsius.value);
6    var F =Math.round(C*9/5+32);
7    document.f.fahrenheit.value=F;
8     }
9
10 function F2C() {
11 var F = eval(document.f.fahrenheit.value);
12 var C = Math. round((F - 32 ) * 5/9);
13 document.f.celsius.value=C;
14  }
15 </script>
16 </head>
```



Obr. 6.3 javascriptové funkce převodu teploty a výsledný převod teplot v prohlížeči

Tab. 6.5 Matematické konstanty

Konstanta	Popis	Konstanta	Popis
Math.E	E, základ logaritmu	**Math.LOG10E**	desítkový logaritmus z e
Math.PI	Pí, Ludolfovo číslo	**Math.LOG2E**	dvojkový logaritmus z e
Math.LN10	přirozený logaritmus z deseti	**Math.SQRT2**	odmocnina ze dvou
Math.LN2	přirozený logaritmus ze dvou	**Math.SQRT1_2**	odmocnina z jedné poloviny

Druhým příkladem pro vyzkoušení javascriptových příkladů bude jednoduchý kalkulátor. Nadefinujeme si dva textboxy pro aritmetické hodnoty a jeden textbox pro zobrazení výsledků. Dále přidáme do formuláře dvě tlačítka – jedno pro sčítání a druhé pro odečítání. Tlačítka budou reagovat na kliknutí a pomocí události *onClick* volat funkce které si nadefinujeme v hlavičce dokumentu.

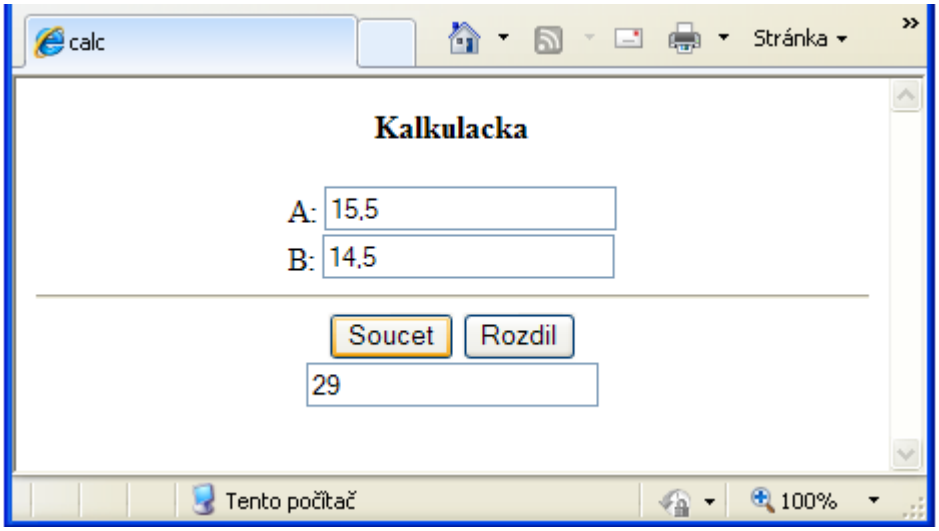
```
14 <body>
15 <center>
16 <b>Kalkulacka</b>
17 <br>
18 <form name="formular">
19 A: <INPUT type="text" name="a" ><BR>
20 B: <INPUT type="text" name="b" ><BR>
21 <hr>
22 <INPUT TYPE="button" value="Soucet" onClick="Secti()">
23 <INPUT TYPE="button" value="rozdil" onClick="Odecti()"> <br>
24 <INPUT TYPE="TEXT" name="vysledek">
25
26 </FORM>
27 </center>
28 </body></html>
```

Obr. 6.4 Formulář kalkulátoru

Funkce pro sčítání a odčítání jsou realizovány podle zmíněné tečkové konvence objektů a podobjektů. Protože ale číselnou hodnotu načítáme z textboxu, musíme textovou proměnnou převést na číslo. K tomu slouží metoda *parseFloat*, která převádí hodnotu na reálné číslo. Pokud by nám stačilo pracovat s celými čísly, použili bychom metodu *parseInt*. Těmito metodami opět javascript ukazuje, že ačkoliv datové typy nedeklaruje, ve skutečnosti s nimi počítá.

```
1 <HTML>
2 <head> <title>calc</title>
3 <script type="text/javascript">
4 function Secti() {
5 document.formular.vysledek.value=
6 parseFloat(document.formular.a.value) + parseFloat(document.formular.b.value) ;
7 }
8 function Odecti() {
9 document.formular.vysledek2.value=
10 parseFloat(document.formular.a.value) - parseFloat(document.formular.b.value) ;
11 }
12 </script>
13 </head>
```

Obr. 6.5 Funkce kalkulátoru



Obr. 6.6 Výsledný kalkulátor v prohlížeči

Třetí příklad bude využívat objektu *Date*, který poskytuje nejen údaje datumu ale také času. V tabulce vidíme metody, které můžeme využívat. Přesto, že se jedná o objekt data a času, metody vracejí převážně jen číslo od určité startovní hodnoty, nečekejme tedy, že by metoda vrátila textovou informaci.

Tab. 6.6 Metody objektu Date

Metoda	Popis	Metoda	Popis
getYear()	Rok dvěma číslicemi	**getTime()**	Počet milisekund od 1.1.1970
getFullYear()	Rok čtyřmi číslicemi	**getHours()**	Počet hodin od půlnoci
getMonth()	Měsíc (leden = 0)	**getMinutes()**	Počet minut od začátku hodiny
getDate()	Číslo dne v měsíci	**getSeconds()**	Počet sekund od začátku minuty
GetDay()	Číslo dne v týdnu (neděle = 0)	**getMiliseconds()**	Počet milisekund od začátku sekundy

Protože v našem příkladu chceme zobrazovat přesné datum navíc v češtině, musíme udělat malou přípravu. Především nadefinovat pole dnů v týdnu, které navíc začínají nedělí. Druhé pole bude obsahovat názvy měsíců. Založíme si také proměnnou objektu *Date*.

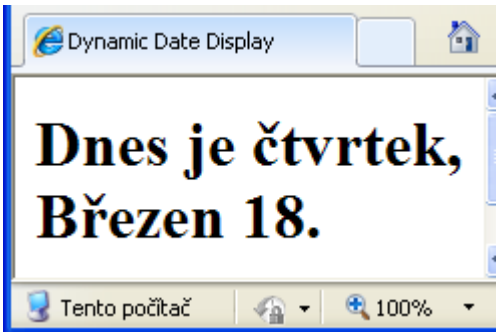
```
<HTML><HEAD>
    <TITLE>Dynamic Date Display</TITLE>
    <SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT">

    dayName = new Array ("Neděle","Pondělí","Uterý","Středa","čtvrtek","Pátek","Sobota")
    monName = new Array ("Leden", "Unor", "Březen", "Duben", "Květen",
    "Cerven", "Cervenec", "Srpen", "Září", "Říjen", "Listopad", "Prosinec")

    now = new Date

    </SCRIPT>
</HEAD>
```

Obr. 6.7 Definice textových polí pro zobrazení aktuálního data

Pro textový výstup zvolíme metodu *dokument.write* ve které můžeme textové proměnné míchat z html značkami a řetězci v uvozovkách. Metody objektu *Date*, konkrétně *getDay*, *getMonth* a *getDate* vracejí pouze celé číslo, v našem případě to bude index pole dnů v týdnu a měsíce v roce. Pro pochopení pole *dayName[]* je prvním prvkem pole *dayName[0]* neděle,druhým prvkem *dayName[1]* pondělí, *atd.* Důležitým faktem tedy je, že v javascriptu se indexuje pole od nuly podobně jako například v jazyce C, tudíž pole o deseti prvcích má prvky pole[0] až pole[9].

```
13 <BODY>
14 <SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT">
15
16     document.write("<H1>Dnes je " + dayName[now.getDay()] + ", " +
17     monName[now.getMonth()] + " " + now.getDate() + ".<\/H1>")
18     document.write(now.getMonth())
19
20 </SCRIPT>
21 </BODY></HTML>
```

Obr. 6.8 Výpis data podle hodnot prvků pole



Obr. 6.9 Textový výpis data v prohlížeči

6.5 Praktické příklady

Tak jako jsme si u kaskádového pozicování a tvorby CSS šablon ukázali jakým způsobem je možné využívat již připravených hotových řešení v našich aplikacích, stejně tomu bude i s javascriptovými příklady. Rovněž v tomto případě existuje stovky hotových skriptů, které můžeme využít ve svých aplikacích. Webové stránky pracují přes různý informační obsah na podobných principech, takže věřte, že to, co se hodí do stránek vám, již určitě někdo před vámi také potřeboval. V této kapitole si ukážeme, jakým způsobem můžeme využít hotové skripty a kde je můžeme najít.

Nejprve si ukážeme velmi oblíbený prvek webových stránek a tím jsou hodiny. V první ukázce si do html stránky přidáme vlastní skript pro zobrazení digitálních hodin, v ukázce druhé bude volat vzdálený skript, který bude zobrazovat dokonce analogové zobrazení hodin včetně sekundové ručičky.

Pro zobrazení digitálních hodin vytvoříme jednoduchý element se selektorem `ID="Clock"`, na který budeme hodiny odkazovat. Hodiny budou realizovány funkcí *ShowTime()*, která bude spuštěna po načtení html stránky do prohlížeče. Budeme ji tedy volat událostí *OnLoad* v tělě dokumentu.

```
30   <BODY OnLoad="ShowTime();">
31     Aktualní čas<BR>
32     <B ID="Clock">00:00:00</B>
33   </BODY>
34 </HTML>
```

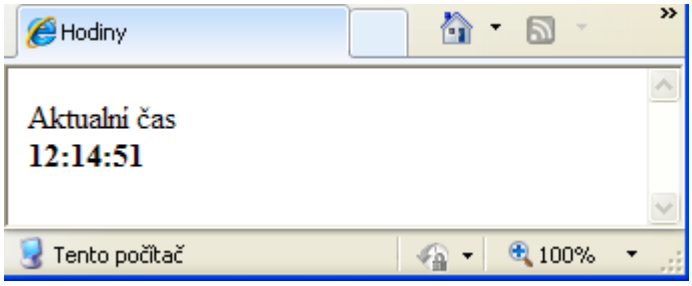
Obr. 6.10 Zdrojový kód umístění digitálních hodin do těla dokumentu

Funkce *ShowTime()* bude definována v hlavičce html stránky. Pro zobrazení času potřebujeme znát hodnotu hodin, minut a sekund. To zjistíme metodami objektu *Date*. Zobrazovaný řetězec poskládáme ze všech získaných údajů, avšak nastává drobný problém s hodnotou minut a sekund, v případě, že bude menší než deset. V tomto případě musíme ještě zajistit dvouciferné zobrazení pomocí uvozující nuly na místě první číslice (např. 16:5:20 musíme převést na 16:05:20). K tomu použijeme ternální operátor **?**, který pomůže vyhodnotit uvedenou podmínku. Protože se čas mění co sekundu, musíme zajistit obnovení funkce *ShowTime()*. To se v javascriptu obecně provádí funkcí *setTimeout()*, které však musíme parametr nastavit na 1000, neboť tento parametr je v milisekundách.

```
<HTML>
  <HEAD>
    <TITLE> Hodiny </TITLE>
    <SCRIPT type="text/javascript"  Language="JavaScript">
      <!--
      function ShowTime(){
        // Zjistí se aktuální čas
        var D = new Date;
        // Z časového údaje se zjistí počet
        // hodin, minut a sekund
        var H = D.getHours();
        var M = D.getMinutes();
        var S = D.getSeconds();

        // Vytvoří se řetězec obsahující celý čas
        // U minut a sekund se přidají případné
        // uvozující nuly (pokud jich je méně jak 10)
        var StrTime = H + ":" +
          (M<10 ? "0"+M : M) + ":" +
          (S<10 ? "0"+S : S);
        // Změní se obsah značky daného jména
        Clock.innerText = StrTime;
        // Aktualizace hodin se provede za sekundu
        setTimeout ("ShowTime()", 1000);
      }
      // -->
    </SCRIPT>
  </HEAD>
```

Obr. 6.11 Zdrojový kód funkce, která zajišťuje běh digitálních hodin

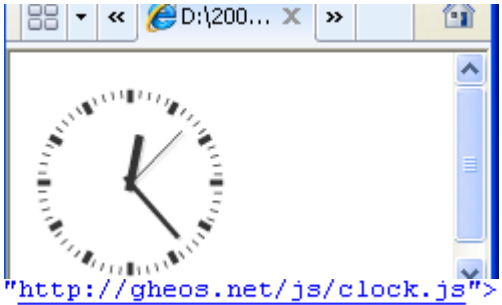
Zdrojový kód funkce *ShowTime()* vidíme na obrázku 6.11, kde je přehledně komentován. Digitální hodiny tak máme hotové a jejich umístění si můžeme určit sami.



Obr. 6.12 Výsledné zobrazení digitálních hodin v prohlížeči

Zobrazení analogových hodin je oproti digitálním pochopitelně složité, neboť kromě času musíme ještě zobrazit grafickou část ciferníku. Proto s oblibou využíváme již hotové skripty.

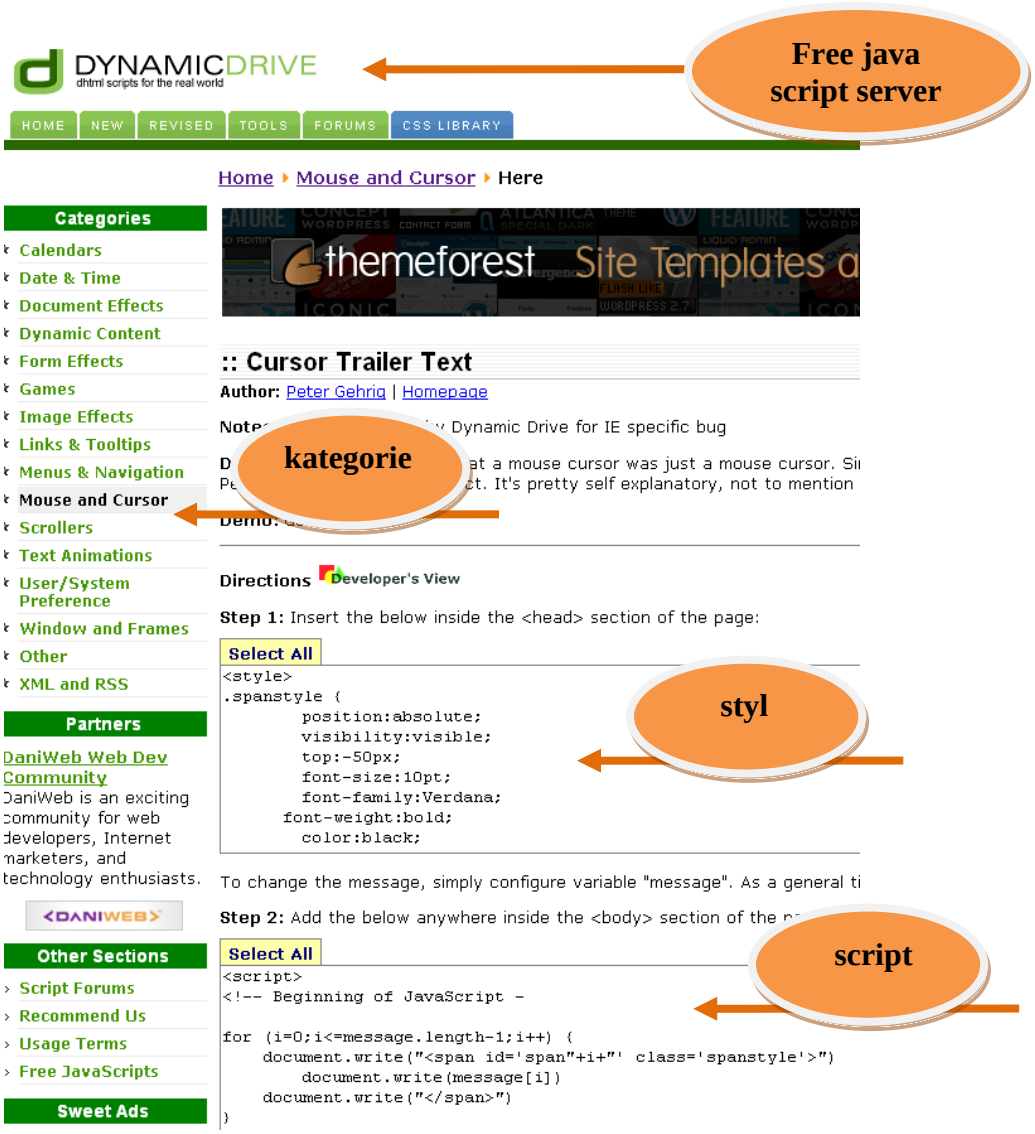
```
<html>
<head>
<script type="text/javascript">
var clocksize='100';
</script>
</head>
<body>
<script type="text/javascript" SRC="http://gheos.net/js/clock.js">
</script>
</body></html>
```



Obr. 6.13 Volání vzdáleného skriptu a výsledné analogové hodiny v prohlížeči

Analogové hodiny tedy budeme volat vzdáleně ze serveru *gheos.net*, kde je script umístěn v souboru *clock.js*. O to všechno to bude jednodušší. Našim úkolem je pouze nastavit velikost zobrazovaných hodin a na skript se odkázat paramentrem *SRC*. Jak jednoduché, že? Mějme ale na paměti, že při výpadku vzdáleného serveru se skriptem se na našich stránkách hodiny nezobrazí. Proto je vhodné volat vzdálené skripty pouze u spolehlivých serverů a nebo skripty raději umísťovat přímo do zdrojového kódu našich stránek. Tento způsob si ukážeme nyní.

Pro umístění hotových skriptů do našich stránek platí určitá pravidla. Obecně vyhledání skriptu a stažení zdrojových kódů a potřebných objektů jako jsou obrázky apod. je stejné jako v předchozích případech, kdy jsme ukazovali hotová CSS řešení. U javascriptu je však rozdíl v tom, že pokud nějaký script umístíme do hlavičky stránek, obvykle v nějaké funkci, musíme zajistit volání této funkce. Nejčastěji to bude opět na událost *OnLoad* v těle dokumentu. Pokud již ale na tuto událost voláme jinou funkci, je třeba volané funkce oddělit středníkem. Na webových serverech věnovaných javascriptu je mnoho zdařilých skriptů, které jistě použijete. Nejčastěji se kromě hodin, datumových, kalendářových a různých výpočetních skriptů ještě využívají kurzory myši a stavový pruh prohlížeče, jsou však pokročilé skripty, kdy můžete implementovat do stránek třeba počítačovou hru. V této kapitole již nebudeme uvádět seznamy URL adres nejvhodnějších serverů s javascriptovou tematikou. Vyhledání vhodných skriptů necháme na rozvaze studenta, vždyť zadat klíčová slova jako *free java script* do vyhledávače google umí každý.



Obr. 6.14 Výběr všech součástí skriptu pro použití ve vlastních stránkách

Ke kapitole 6 je přiřazena demonstrační animace č.3

Druhá část animace č. 3 obsahuje praktickou ukázku tvorby, umístění a využití javascriptu v html stránce.

Ke kapitole 6 je přiřazen demonstrační program č.4

Program č. 4 demonstruje ukázku použití užitečných javascriptů ve webových stránkách.

Shrnutí kapitoly

Javascript je skriptovací jazyk na straně klienta, to znamená, že webový server nemá žádnou informaci o tom co se na klientském počítači děje. Všechny akce, které provádí script se dějí na klientské straně.

V interakci s html jazykem dokáže reagovat na události jako klikání myší, změna polohy objektů apod. Pomocí něj můžeme validovat data z formulářů na straně klienta. Dále dokáže vytvořit dynamický text v HTML stránce, přizpůsobovat a měnit HTML elementy. Mimo jiné také umožňuje číst a zapisovat proměnné cookies, detekovat klientův prohlížeč a další užitečné vlastnosti.

Proměnné se obvykle deklarují a jsou určitého datového typu. Slovo obvykle je tu záměrně protože právě pro javascript tato charakteristika neplatí. Proměnné můžeme, ale nemusíme deklarovat a datový typ k nim nepřiřazujeme, potřebujeme pouze vědět, zda-li se jedná o řetězec, logickou hodnotu či číslo a o tom rozhodnou uvozovky.

Javascript je case sensitive, to znamená, že na rozdíl od HTML rozlišuje velká a malá písmena. Proměnná *Jmeno* je tedy odlišná od proměnné *jmeno*. Javasript je částečně objektový jazyk, nevyužívá možnosti objektově orientovaného programování jako pokročilé OOP jazyky. U javaskriptu se pod pojmem objektový chápe jisté uspořádání systému a přístup k objektům pomocí metod a vlastností. K objektům a podobjektům se přistupuje tečkovou konvencí rovněž metoda objektu je volána způsobem objekt.metoda(). Pokud budeme mít v dokumentu formulář obsahující textbox s hodnotou proměnné, pak se podle tečkové konvence na tuto proměnnou budu odkazovat:

dokument.formulář.textbox.proměnná.její_hodnota

Nejčastějšími objekty, se kterými javascript pracuje je object *document*, *Date* a *Math*. Javascript je jazyk vycházející ze syntaxe Javy či jazyka C. Kromě metod objektů často využíváme určité příkazové konstrukce. Ke každému jazyku programovacímu či skriptovacímu patří programové konstrukce jako porovnávání, rozhodování, větvení, cykly a funkce. K této kapitole je připraven program se zdrojovými kódy příkazových konstrukcí, které jsou demonstrovány na vzorových příkladech.

Úkol k řešení 6.1 – Jednoduchý kalkulátor v javascriptu

Na základě příkladu z učebního textu vložte do formuláře další tlačítka, která budou mít funkčnost násobení a dělení a také odmocniny z proměnné vložené do textboxu pro proměnnou *a*. Použijte opět objekt Math.

Úkol k řešení 6.2 – Objekt Datum

Využijte metody objektu datum pro zobrazení aktuálního data ve vašich stránkách.

Úkol k řešení 6.3 – Analogové hodiny

Obdobným postupem jako u příkladu v učebním textu vložte do vašich stránek analogové hodiny v javascriptu.

Úkol k řešení 6.4 – Kurzor myši

Najděte vhodný javascript a implementujte do svých stránek textový efekt běžící textu za kurzorem myši.

Kontrolní otázka 6.1

Jaké základní vlastnosti a schopnosti má javascript?

Kontrolní otázka 6.2

Co pomocí javascriptu nelze provést?

Kontrolní otázka 6.3

Co znamená operátor modulo?

Kontrolní otázka 6.4

Definujte výsledek matematických operací 5%3 a 6%2.

Kontrolní otázka 6.5

Co je to událost a jaké události můžeme obsluhovat?

Kontrolní otázka 6.6

Co je to objektový model dokumentu?

Kontrolní otázka 6.7

Jaké objekty využíváme v javascriptu nejčastěji?

Kontrolní otázka 6.8

Co vrací metoda getMonth() objektu Date?

7. ZÁKLADNÍ ELEMENTY KOMUNIKACE V POČÍTAČOVÝCH SÍTÍCH

Čas ke studiu: 2 hodiny

Cíl Po prostudování této kapitoly budete umět

- definovat 4 základní elementy komunikace v síti
- rozdělit typy zařízení na síťová a koncová
- popsat typy komunikačních médií a určit jejich vhodné použití
- rozdělit počítačové sítě podle několika kritérií
- definovat pojem protokol a pochopit jeho význam v síťové komunikaci

Výklad

Druhá polovina semestru a také učebního textu je věnována základům počítačových sítí. Abychom pochopili podstatu síťové komunikace, je třeba vysvětlit architekturu síťového modelu a přiblížit si práci na jeho jednotlivých vrstvách. Úkolem úvodní kapitoly je definovat základní pojmy, se kterými se budeme neustále setkávat. Těmito pojmy není nic jiného než síťová a koncová zařízení, komunikační média a protokoly.

7.1 Základy komunikace v počítačové síti

Základem každé komunikace mezi dvěma objekty je souhrn prostředků a pravidel, která nám umožňují komunikaci provést. Jako příklad si můžeme uvést běžnou komunikaci mezi dvěma lidmi, se kterou se setkáváme dennodenně. Prvním elementem komunikace jsou tedy objekty konverzace, v našem příkladě jsou jimi lidé, v počítačové síti jsou to nějaká hardwarová zařízení. Komunikovat můžeme různým způsobem: telefonicky, prostřednictvím pošty, nebo třeba jen běžným rozhovorem na ulici. Druhým elementem komunikace je tedy přenosové médium naší konverzace. Z uvedeného příkladu vyplývá, že toto médium je mimo jiné voleno na vzdálenosti dvou konverzujících objektů. Mezi další faktory výběru média pak patří jeho cena, kvalita přenosu a další. Třetím faktorem komunikace je to nejdůležitější a tím je samotný obsah sdělení. Právě pro potřebu výměny informací vznikají tyto síťové technologie. Třetí faktor komunikace přímo ovlivňuje faktor čtvrtý, tím je souhrn pravidel komunikace. Můžeme spolu hovořit různými jazyky, můžeme použít v komunikaci šifrovací klíč, jinak spolu hovoříme telefonicky, jinak se oslovujeme v elektronické poště. V klasické listovní poště můžeme poslat zprávu doporučeně, nebo ji pouze vhodit do schránky a spoléhat se, že bude doručena. To vše a mnoho dalšího určuje pravidla komunikace.
Všechny výše popsané elementy klasické lidské konverzace se dají aplikovat na počítačovou síť a tak máme definovány čtyři základní elementy komunikace.

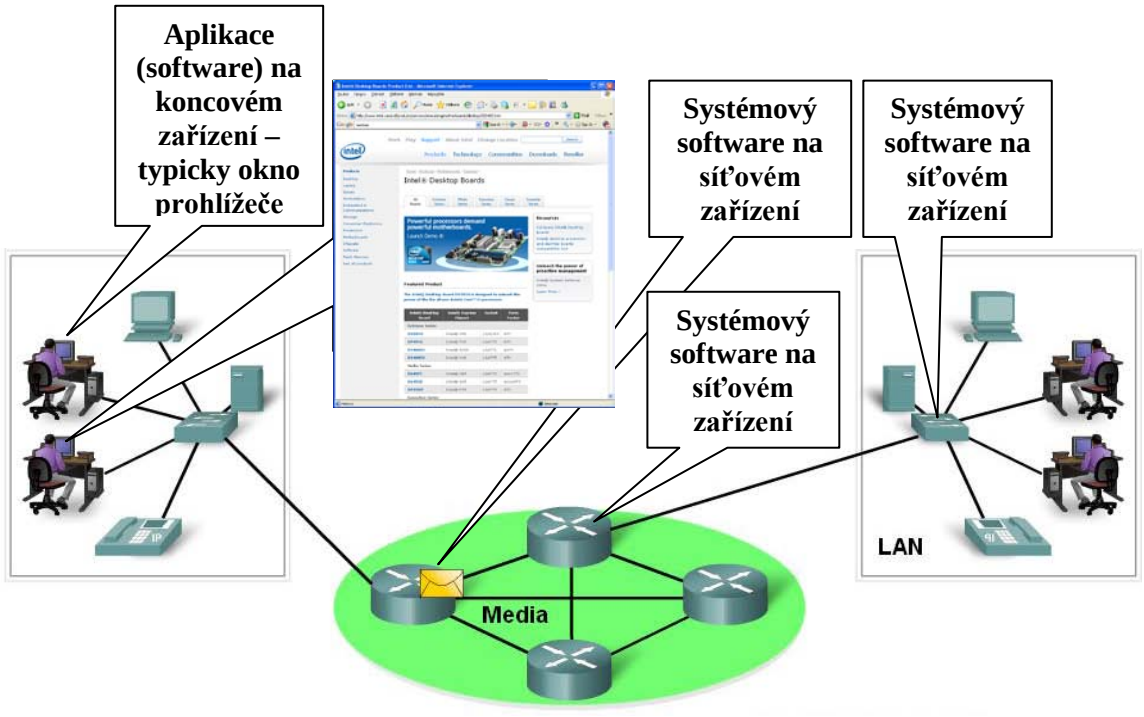
Pojem k zapamatování: 4 elementy komunikace v počítačové síti

Základními elementy komunikace v počítačové síti jsou **zařízení,** které mezi sebou komunikují, komunikační **médium** po kterých jsou zasílány jednotlivé **zprávy** a také souhrn **pravidel** komunikace.

Obr. 7.1 Elementy komunikace v počítačové síti

7.2 Komponenty počítačové sítě

Elementy komunikace popsané v minulé kapitole můžeme chápat jako komponenty počítačové sítě. Typickou komponentou sítě je samotný počítač, který je zároveň počátečním či koncovým prvkem komunikace. Mimo koncových zařízení, která mezi sebou chtějí komunikovat, potřebujeme zařízení, která nám dokáže danou komunikaci zprostředkovat. Tyto zařízení nazýváme aktivní síťové prvky a jsou většinou pod správou síťových administrátorů a běžný koncový uživatel k nim nemá přístup. Tyto prvky si představíme v následujících kapitolách a na několika příkladech zkusíme i základní konfiguraci. Elementy komunikace jsou tedy viditelnými prvky, ale nesmíme zapomenout, že pro spolehlivý průběh komunikace běží na síťových prvcích mnoho procesů a služeb, které se právě starají o chod počítačové sítě.



Obr. 7.2 Komponenty počítačové sítě

Pojem k zapamatování: Komponenty počítačové sítě

Mezi komponenty sítě musíme kromě koncových a síťových zařízení zahrnout také systémový software a služby, které běží na těchto síťových zařízeních.

7.3 Typy zařízení

Zařízení, které používá koncový uživatel se nazývají koncová zařízení (angl. *end devices*). V podstatě jsou to zařízení, se kterými pracuje běžný uživatel, protože jsou to nástroje k jeho každodenní práci. Ostatní zařízení ho nezajímají a v podstatě o nich z bezpečnostních důvodů neměl vědět, a pokud ano, neměl by k nim mít přístup. Mezi koncová zařízení patří:

- PC, Notebook, PDA
- Server, (DB server, Web server, file server)
- Síťová tiskárna
- VoIP telefon
- Kamery
- Další koncová zařízení (čtečky kódu, scannery, atd.)

Tab. 7.1 Označení koncových zařízení

Koncová zařízení		Koncová zařízení	
	**Počítač**		**IP telefon**
	**Notebook**		**Síťová tiskárna**
	**Server**		**PDA**

Síťová zařízení jsou zodpovědná za bezproblémové doručování zpráv v počítačové síti. Mohou se nacházet uvnitř samotné lokální sítě, mezi jednotlivými koncovými zařízeními, propojují ale také mezi sebou více sítí a tvoří tak rozsáhlou počítačovou síť. Tyto zařízení pracují na různých vrstvách síťě (o síťovém modelu budeme hovořit v kapitole 8). To znamená, že každé síťové zařízení má jinou úlohu, je zodpovědné za svoji činnost, pro kterou je do počítačové sítě umístěn. Některá zařízení jsou zodpovědné za to, že pakety se dostanou k cíli tou nejvhodnější cestou, jiná zařízení naopak pouze zesilují signál a přeposílají data. Jiná zařízení mají zase na starosti bezpečnost sítě. Mezi síťová zařízení typicky patří:

- Hub a switch (přepínač)
- Wifi access point
- Router
- Bezpečnostní firewall
- Komunikační servery a modemy

V tabulce 7.1 a 7.2 si ukážeme, s jakými zařízeními se v tomto předmětu můžeme setkat a jakým způsobem budeme jednotlivá zařízení označovat. Tyto symboly budeme používat i v animacích a demonstračním softwaru pro simulaci provozu počítačové sítě.

Pojem k zapamatování: Typy zařízení

Zařízení v počítačové síti dělíme na koncová a síťová. Koncová zařízení využívá uživatel, typicky PC, tiskárna. Síťová zařízení by se měly správně nazývat zprostředkující (ang. i*ntermediary*), neboť zprostředkovávají komunikaci koncovým zařízením v síti.

Tab. 7.2 Označení síťových zařízení

Síťová zařízení		Síťová zařízení	
	**Router**		**Access point**
	**LAN switch**		**Modem**
	**LAN hub**		**Firewall**
	**Wifi router**		**Bridge**

7.4 Komunikační média

Pro zajištění přenosové cesty musíme zvolit vhodné komunikační médium. Jak už bylo řečeno, médium není voleno pouze na základě vzdálenosti, ale rozhoduje o tom celá řada kritérií:

- Cena média a jeho instalace
- Spolehlivost datového přenosu
- Typ přenesených dat (video, hlas, data)
- Očekávané množství přenesených dat
- Požadovaná rychlost
- Typ prostředí, ve kterém se nachází přenosová cesta

Různá komunikační média je používají pro své účely. Každé médium má své výhody a nevýhody a samozřejmě v místě, kde je vhodné instalovat určitý typ média, může jiné médium ač dražší a kvalitnější být zcela nevyhovující. V dnešní době se stále nejvíce setkáváme s komunikačním médiem zvaným kroucená dvoulinka, neboť v administrativních a veřejných prostorách je pro své vlastnosti nejvhodnější. Můžeme se také setkat ale už méně s koaxiálním kabelem nebo dokonce s optickým vláknem, které je ale pro svou vysokou cenu implementován spíše na vysokorychlostní páteřní síť.
S rostoucí popularitou bezdrátového připojení nelze opominout ani tento typ přenosu, který využívá jako komunikační médiu vzduch.

Tab. 7.3 Komunikační média a jejich konektory

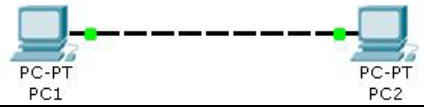
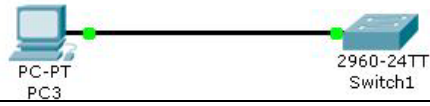
Médium	Kabel	Konektor
Kroucená dvoulinka		
Koaxiální kabel		
Optické vlákno		
Wireless		

Pojem k zapamatování: Přenosové médium a jeho signál

V metalickém provedení přenosového média je signál generován elektrickými impulsy podle nějakého předepsaného kódování. Při optickém přenosu jsou data přenášena světelnými pulsy. Bezdrátový přenos je realizován pomocí elektromagnetických vln.

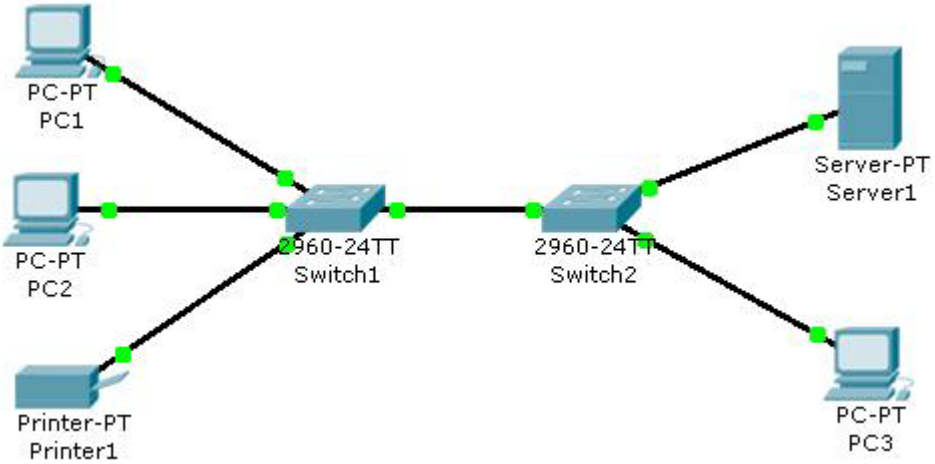
Následující příklad nás připravuje na první cvičení propojení koncových a síťových zařízení. Demonstruje typy zapojení. V této chvíli ještě nebudeme vysvětlovat, která zařízení můžeme propojovat mezi sebou přímým a která křížovým kabelem. To je podrobně popsáno v kapitole věnované fyzické vrstvě. Zatím nám postačí definovat čtyři druhy zapojení, se kterými budeme během semestru pracovat.

Příklad 7.1 – Propojení prvků zařízení v demonstračním programu

	Propojení dvou zařízení kříženým kabelem.
	Propojení dvou zařízení přímým kabelem.
	Propojení PC a síťového zařízení konzolovým kabelem pro konfiguraci.
	Propojení dvou WAN routerů seriovým kabelem.

Příklad 7.2 – propojení několika koncových zařízení do přepínačů

Na obrázku vidíme přímo ukázku zapojení dvou PC a síťové tiskárny do switche 1 a servru a PC do switche 2. Oba switche jsou mezi sebou rovněž propojeny. Ukázka je provede v programu Packet Tracer společnosti Cisco a obsahem animace a příkladů na cvičení bude sestavování obdobných jednoduchých topologií. Konfiguraci a nastavení IP adres jednotlivých zařízení budeme řešit v dalších kapitolách.

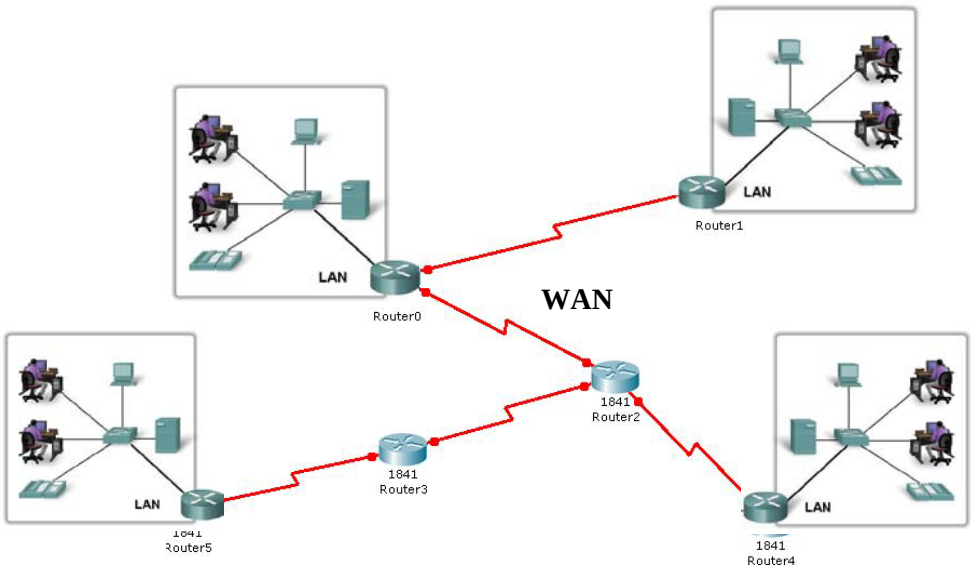


7.5 Rozdělení sítí

Počítačové sítě můžeme rozdělovat dle mnohých kritérií. Od jednoduchých kritérií daných počtem uživatelů, rychlostí sítě, typu přenosového média a dalších faktorů až po škálovatelná kritéria, která dělí sítě na veřejné, privátní, pronajaté atd. Každé kritérium dělení sítí je důležité pro její samotnou realizaci neboť charakteristika sítě a důvod jejího použití je zásadní pro její počáteční návrh a odvíjí se od ní veškeré prostředky implementace. Obsahem kapitoly ale nebude probírat detaily speciálních typů sítí, pouze si sítě rozdělíme na lokální a rozsáhlé a rozdělíme si je podle možnosti připojení.

Pojem k zapamatování: Faktory rozdělení sítí

O typu konkrétní sítě jednoznačně rozhoduje faktor geografického pokrytí, to znamená jak velkou část je třeba sítí pokrýt (od jedné místnosti až po tisíce kilometrů). Z tohoto faktoru vychází také důležitý údaj o počtu uživatelů a množství přenesených dat.



Obr. 7.3 Rozsáhlá počítačová síť propojující několik lokálních sítí

Lokální počítačová síť obvykle pokrývá male území a zajišťuje podmínky pro práci lidem jedné organizační složky s nějakou hierarchií, definuje své požadavky na služby sítě a typ použití počítačové sítě.
Pokud velká organizace spravuje vice lokálních sítí na různých geografických územích, jedná se o rozsáhlou počítačovou síť, a propojení mezi jednotlivými lokálními sítěmi zajišťuje nějaký externí subjekt, kterému se říká *telekomunikační provider*. Telekomunikační provider je zodpovědný za provoz rozsáhlé sítě od určitých bodů propojení lokálních sítí.

Pojem k zapamatování: LAN a WAN

Lokální počítačová síť pokrývá jedno souvislé území a je plně pod správou jedné organizace včetně poskytovaných služeb a zajištění bezpečnosti sítě. Rozsáhlá počítačová síť propojuje sítě lokální a od těchto bodů připojení je pod plnou správou telekomunikačního providera.

Pojem k zapamatování: Internet vs. Intranet

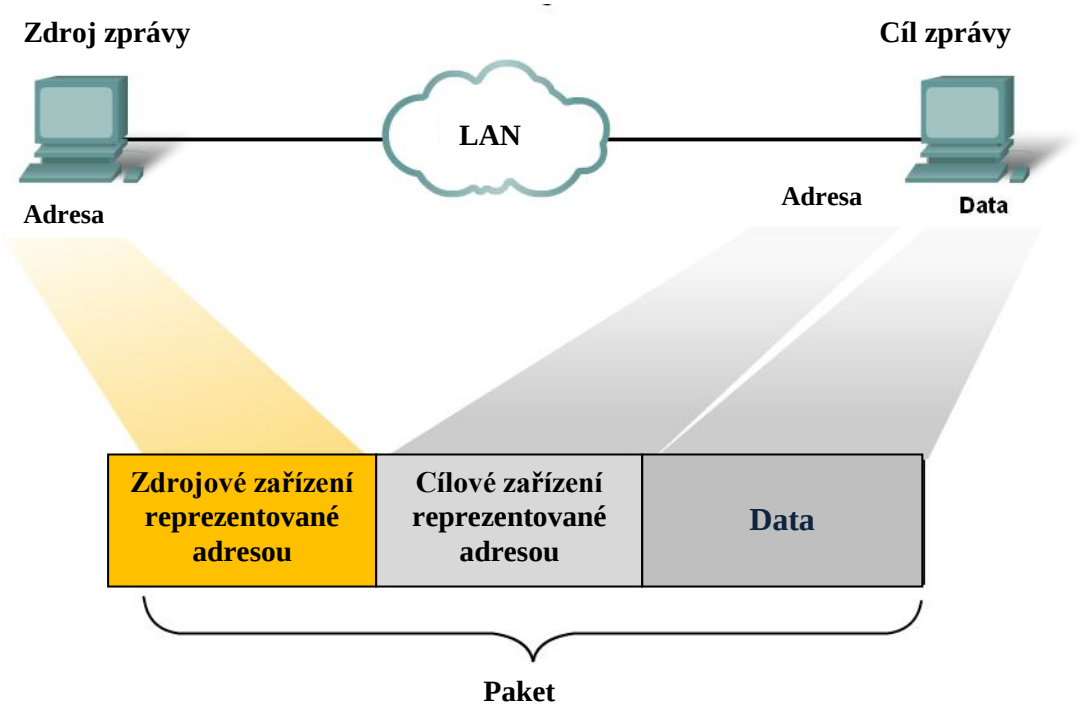
S pojmy LAN a WAN nepřímo souvisí pojem Internet a Intranet. Internet je celosvětová veřejná síť, ke které se připojíme opět prostřednictvím *providera*. Naopak Intranet je privátní síť pod správou organizace přístupná pouze pro členy lokální sítě právě z důvodu bezpečnosti a z okolních sítí je nepřístupná.

Tab. 7.4 Možnosti připojení k rozsáhlým počítačovým sítím

Možnosti připojení WAN			
Soukromé			**Veřejné**
Pevné připojení	*Přepojované*		*Internet*
Pronajatá linka	Přepojování okruhů ISDN, PSTN	Přepojování paketů Frame Relay, ATM	Širokopásmové DSL, Kabel, Wireless

7.6 Komunikační protokol

V reálném životě existuje mnoho způsobů komunikace. Aby se mezi sebou domluvily dva objekty, musí nejprve zvolit společný způsob komunikace, kterému rozumí obě strany. Tento zvolený způsob komunikace určuje souhrn pravidel, jímž se komunikace řídí. Stejně tak mezi dvěma počítači musí existovat společný formát komunikace, tak aby vyslanou zprávu z jednoho koncového zařízení obdrželo druhé koncové zařízení a hlavně aby této zprávě porozumělo.
V dnešní době existuje mnoho typů počítačových sítí, mnoho různých koncových zařízení s odlišnými operačními systémy. Mnoho sítí je navrženo na zcela odlišných principech komunikace, sítě jsou mezi sebou propojeny mnoha typy komunikačních médií. To vše museli vzít v potaz odborníci, kteří navrhli celou řadu standardů a architekturu síťového modelu.



Obr. 7.4 Komunikace dvou zařízení pomocí protokolu, jehož jednotkou je paket

Jak už tedy bylo řečeno souhrn pravidel komunikace je určující pro konverzaci dvou koncových zařízení. Souhrn pravidel je realizován komunikačními protokoly, které se starají o chod konverzace. Tato konverzace je zajištěna vytvořenou přenosovou cestou interakcí mnoho protokolů, které pracují

na architektuře síťového modelu, která bude představena v další kapitole. Informační jednotkou, která je po vytvořené síťové cestě předávána je paket. Formát paketu je na každé vrstvě modelu odlišný, dokonce různé specifikace protokolů na stejné vrstvě síťového modelu, avšak jiné technologie mohou mít odlišné formáty paketů.
Zodpovědnost spolehlivé komunikace je právě v interakci protokolů na jednotlivých vrstvách tak aby si jednotlivá zařízení po síťové cestě vždy rozuměli.

Pojem k zapamatování: Komunikační protokol

Úkolem komunikační protokolu je zajistit správné doručení zprávy ze zdrojového do cílového zařízení. Komunikační protokoly mají na starost:

- Zajistit vhodný formát a strukturu zprávy
- Vytvořit cestu přes správná síťová zařízení
- Zajistit počátek a konec konverzace
- Detekovat možné příčiny chyb a informovat o tom příslušná zařízení

Na obrázku 7.4 vidíme obecný paket, který je vyslán z jednoho počítače na druhý. Formát paketu se po síťové cestě bude měnit, právě podle toho jaký protokol ho bude mít na starost. Obecný tvar však vždy vypadá takto, paket musí obsahovat adresu zdrojového počítače, cílového počítače a vlastní data pro přenos. Každá konverzace mezi dvěma koncovými zařízeními s sebou nese stovky či tisíce paketů, které mají shodnou počáteční a cílovou adresu avšak jiná data a úkolem protokolů je tyto data dovést správně do cíle.

Ke kapitole 7 je přiřazena demonstrační animace č. 4

Animace č. 4 obsahuje ukázku pracovní plochy programu Packet Tracer, ukázku jednotlivých síťových prvků a jejich vkládání na pracovní plochu, ukázku propojovacích kabelů, propojení prvků různým typem propojovacího kabelu, ukázku portů jednotlivých zařízení s důrazem na množství portů u switche, možnosti editace prvku, zobrazení portů, vypnutí a zapnutí routeru, konfigurační režim síťového i koncového zařízení.

Shrnutí kapitoly

Základem každé komunikace mezi dvěma objekty je souhrn prostředků a pravidel, která nám umožňují komunikaci provést. V počítačové síti jsou to **zařízení,** které mezi sebou komunikují, komunikační **médium** po kterých jsou zasílány jednotlivé **zprávy** a také souhrn **pravidel** komunikace.

Elementy komunikace můžeme chápat jako komponenty počítačové sítě. Typickou komponentou sítě je počítač, který je zároveň počátečním či koncovým prvkem komunikace. Pro realizaci konverzace koncových zařízení potřebujeme tuto komunikaci zprostředkovat aktivním síťovým zařízením, které je většinou pod správou síťových administrátorů.

Mezi **koncová** zařízení patří: PC, notebook, PDA, server, síťová tiskárna VoIP telefon, kamera a další. **Síťová** zařízení jsou zodpovědná za bezproblémové doručování zpráv v počítačové síti. Mohou se nacházet uvnitř samotné lokální sítě, mezi jednotlivými koncovými zařízeními, propojují ale také mezi sebou více sítí a tvoří tak rozsáhlou počítačovou síť. Mezi síťová zařízení typicky patří: hub a switch, wifi access point, router, firewall a komunikační servery a modemy.

Pro zajištění přenosové cesty musíme zvolit vhodné **komunikační médium**. O jeho volbě rozhoduje celá řada kritérií: Cena média a jeho instalace, spolehlivost datového přenosu, typ přenesených dat a jeho očekávané množství, požadovaná rychlost, typ prostředí ve kterém se nachází přenosová cesta a samozřejmě vzdálenost.

Počítačové sítě můžeme rozdělovat dle mnohých kritérií. O typu konkrétní sítě jednoznačně rozhoduje faktor geografického pokrytí, to znamená jak velkou část je třeba sítí pokrýt (od jedné místnosti až po tisíce kilometrů). Z tohoto faktoru vychází také důležitý údaj o počtu uživatelů a množství přenesených dat. **Lokální počítačová síť** pokrývá jedno souvislé území a je plně pod správou jedné organizace včetně poskytovaných služeb a zajištění bezpečnosti sítě. **Rozsáhlá počítačová síť** propojuje sítě lokální a od těchto bodů připojení je pod plnou správou telekomunikačního *providera*.

Souhrn pravidel je realizován komunikačními protokoly, které se starají o chod konverzace. Tato konverzace je zajištěna vytvořenou přenosovou cestou interakcí mnoho protokolů síťového modelu. Informační jednotkou, která je po vytvořené síťové cestě předávána je **paket**. Formát paketu je na každé vrstvě modelu odlišný a zodpovědnost spolehlivé komunikace je právě v interakci protokolů na jednotlivých vrstvách tak aby si jednotlivá zařízení po síťové cestě vždy rozuměli. Úkolem komunikační protokolu je zajistit správné doručení zprávy ze zdrojového do cílového zařízení. **Komunikační protokoly** mají na starost: zajistit vhodný formát a strukturu zprávy, vytvořit cestu přes správná síťová zařízení, zajistit počátek a konec konverzace a také detekovat možné příčiny chyb a informovat o tom příslušná zařízení.

Úkol k řešení 7.1 – Seznámení s programem Packet Tracer 5.0

V aplikaci Packet Tracer se seznamte s ikonkami jednotlivých zařízení, ať už koncových či síťových (routery, switche) apod. Jednotlivé zařízení z této kolekce můžete přesouvat na pracovní plochu. Také si procvičte kategorii propojení těchto prvků (ikonka *Connections*). Pokuste se jednotlivá zařízení mezi sebou propojit.

Úkol k řešení 7.2 – Propojení zařízení v programu Packet Tracer 5.0 pomocí různých typů kabelu

Dle obrázku z příkladu 7.1 vytvořte propojení mezi zařízeními čtyřmi typy propojení: přímým, křížovým, konsolovým a sériovým WAN kabelem.

Úkol k řešení 7.3 – Procvičení propojení zařízení v programu Packet Tracer 5.0

Navrhněte a realizujte v programu topologii znázorněnou na obrázku v příkladu 7.2.

Kontrolní otázka 7.1

Jaký je rozdíl mezi základním elementem komunikace a komponentou počítačové sítě?

Kontrolní otázka 7.2

Jaké znáte typy koncových zařízení?

Kontrolní otázka 7.3

Jakou funkci zajišťuje síťové zařízení?

Kontrolní otázka 7.4

Jaké komunikační média počítačových sítí znáte?

Kontrolní otázka 7.5

Jakým způsobem je přenášen signál po komunikačním médiu?

Kontrolní otázka 7.6

Co je to telekomunikační provider?

Kontrolní otázka 7.7

Jakým způsobem můžeme rozdělit typy sítí?

Kontrolní otázka 7.8

Co je to komunikační protokol?

Kontrolní otázka 7.9

Jakým způsobem komunikují mezi sebou dvě zařízení v počítačové síti?

8. ISO OSI A TCP/IP MODEL

Čas ke studiu: 2 hodiny

Cíl Po prostudování této kapitoly budete umět

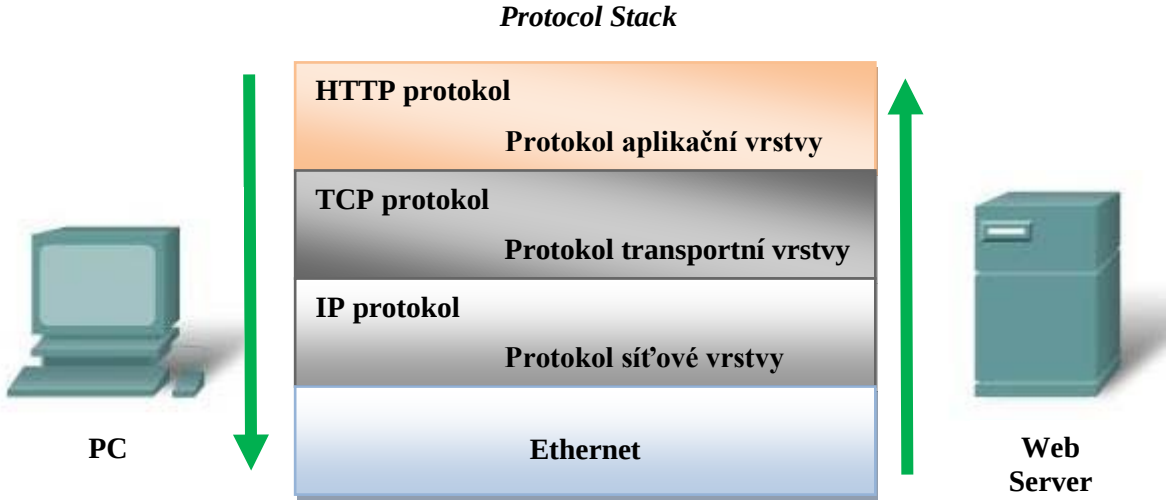
- definovat podstatu komunikace mezi dvěma koncovými zařízeními
- popsat úkoly jednotlivých vrstev síťového modelu
- srovnat ISO - OSI a TCP/IP model
- popsat proces odesílání dat v paketech
- definovat formát dat na jednotlivých vrstvách modelu
- popsat proces zabalení a rozbalení datového rámce

Výklad

V předchozí kapitole jsme si definovali podstatu komunikace v počítačové síti a nezbytné podmínky pro to, aby spolu mohly dva objekty konverzovat. Závěrem kapitoly bylo definování pojmu protokol, který ustanovuje sadu pravidel vzájemné komunikace. Specifikaci vrstev, na kterých jednotlivé protokoly, které mezi sebou navzájem komunikují, definuje architektura síťového modelu. Tuto architekturu si popíšeme v této kapitole.

8.1 Vzájemná interakce protokolů

Ještě než si představíme síťový model, vysvětlíme si důvod jeho existence. Představme si uživatele u počítače, který právě otevřel webový prohlížeč. Při zapsání adresy URL, od které očekává zobrazení webové stránky, spustil celý proces interakce protokolů.



Obr. 8.1 Protokolový zásobník

Jak bylo uvedeno v minulé kapitole po síťové cestě je mnoho různých zařízení a odlišných technologií a úkolem protokolů je svou vzájemnou spoluprací si porozumět a dovést uživatelův požadavek úspěšně k cíli. Uživatel tedy otevřel webový prohlížeč a vzájemnou spoluprací protokolů musí být zajištěno, aby požadovaná aplikační data, jako například zadané data do webového formuláře se dostala na webový server. V horní vrstvě celého procesu máme tedy aplikaci s jejími daty, v té nejspodnější vrstvě budeme mít nějaký signál reprezentovaný nuly a jedničkami, který doputuje po komunikačním médiu až na webový server.
Vzájemnou interakcí protokolů se vytváří tzv. protokolový zásobník (*Protocol stack*). Ten v sobě obsahuje jednotlivé vrstvy protokolů, hierarchicky právě od aplikačních dat až po zmíněné nuly a jedničky na nejnižší vrstvě.

Pojem k zapamatování: Protokolový zásobník

Protokolový zásobník v sobě obsahuje 4 vrstvy TCP/IP modelu: aplikační, transportní, síťovou a vrstvu síťového rozhraní. Tyto 4 vrstvy zabalují a rozbalují na své cestě datový paket.

Na obrázku 8.1 vidíme protokolový zásobník pro zmíněný příklad komunikace PC s webovým serverem. Z toho vyplývá, že jako aplikační protokol se konverzace účastní protokol *HTTP*, protokolem vrstvy transportní je *TCP*. V jiných případech komunikace po počítačové síti se interakce protokolů účastní jiné protokoly, např. při odesílání e-mailové zprávy je aplikačním protokolem *SMTP*, naopak při jejím přijímání to může být *POP3*, při prohlížení např. streamovaného videa je transportním protokolem *UDP*. Funkci a význam konkrétních typů protokolů si vysvětlíme v dalších kapitolách.

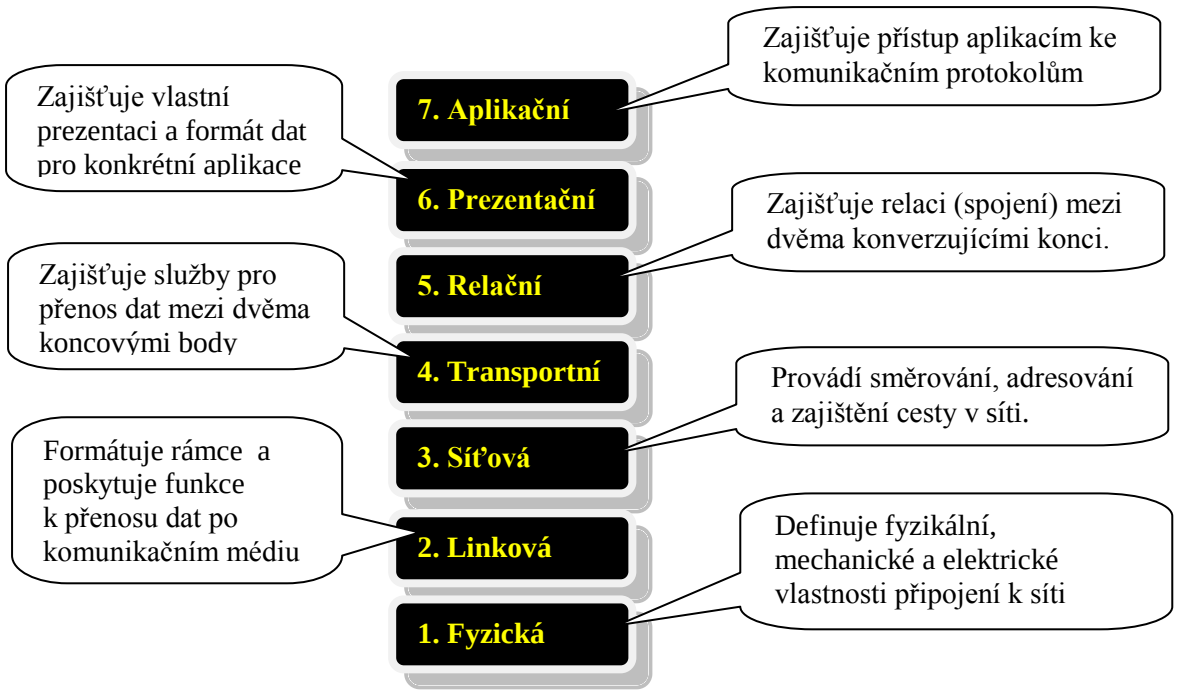
8.2 ISO - OSI model

Při vytváření architektury síťového modelu se používá díky interakcí protokolů právě vícevrstvý model. Tento model definuje úkoly každé vrstvy. To umožňuje výrobcům odlišných technologií vytvářet protokoly podle standardů tak, aby protokoly na jednotlivých vrstvách si dokázaly porozumět. Další výhodou vrstevnatého modelu je fakt, že změna na jedné vrstvě neovlivní vrstvy ostatní, ani nad, ani pod vrstvou, kde ke změně došlo. Mimo koncových zařízení, která mezi sebou chtějí komunikovat, potřebujeme zařízení, která nám dokáže danou komunikaci zprostředkovat. Tyto zařízení nazýváme aktivní síťové prvky a jsou většinou pod správou síťových administrátorů a běžný koncový uživatel k nim nemá přístup. Tyto prvky si představíme v následujících kapitolách a na několika příkladech zkusíme i základní konfiguraci. Elementy komunikace jsou tedy viditelnými prvky, ale nesmíme zapomenout, že pro spolehlivý průběh komunikace běží na síťových prvcích mnoho procesů a služeb, které se právě starají o chod počítačové sítě.

Pojem k zapamatování: Referenční ISO-OSI model

Hlavním úkolem referenčního modelu je definice norem pro účely propojování počítačových sítí. Norma nespecifikuje realizaci systémů, ale uvádí všeobecné principy sedmivrstvé síťové architektury. Popisuje vrstvy, jejich funkce a služby, nikoliv samotné protokoly, jejichž implementace není úkolem síťového modelu.

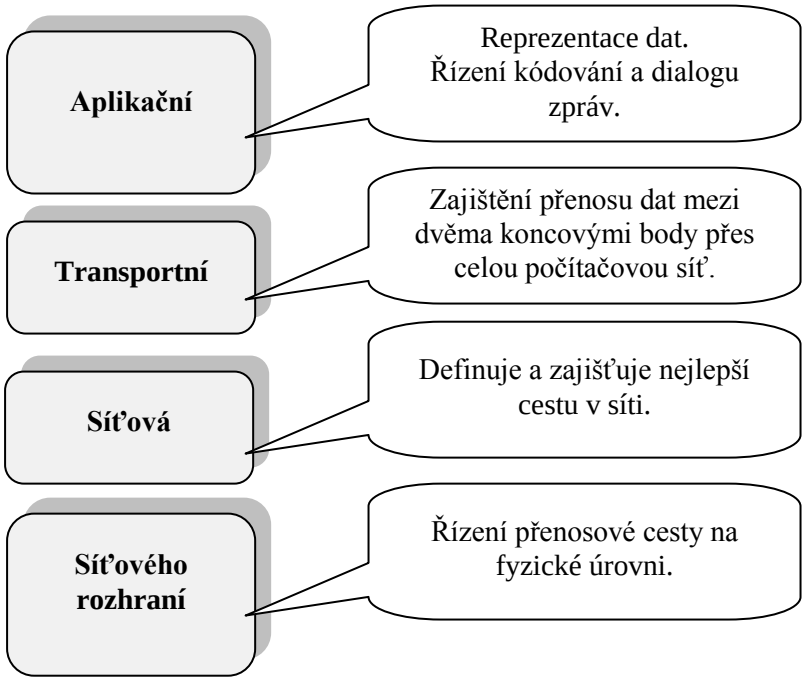
Na obrázku 8.2 je znázorněno všech sedm vrstev architektury ISO – OSI modelu a jsou zde v krátkosti popsány jejich nejdůležitější činnosti. Jednotlivé vrstvy ISO – OSI modelu se dají redukovat podle protokolového zásobníku na čtyři vrstvy TCP/IP modelu. Oba modely si srovnáme a na společných vrstvách budeme definovat jejich charakteristiku, funkčnost a v následujících kapitolách i komunikační protokoly.



Obr. 8.2 Sedmivrstvý ISO - OSI model

8.3 TCP/IP model

Zkráceným, avšak postačujícím síťovým modelem je čtyřvrstvý TCP/IP model. Už v samotném názvu obsahuje transportní a síťový protokol prostředních vrstev, nad nimi je opět vrstva aplikační a na opačné straně vrstva fyzická, která se ovšem nazývá vrstva síťového rozhraní. Tento čtyřvrstvý model koresponduje s protokolovým zásobníkem probraným v úvodu kapitoly a odpovídá mechanismu zapouzdření informačního paketu, který na každé vrstvě modelu TCP/IP zapouzdří nebo rozbalí (podle směru komunikace).



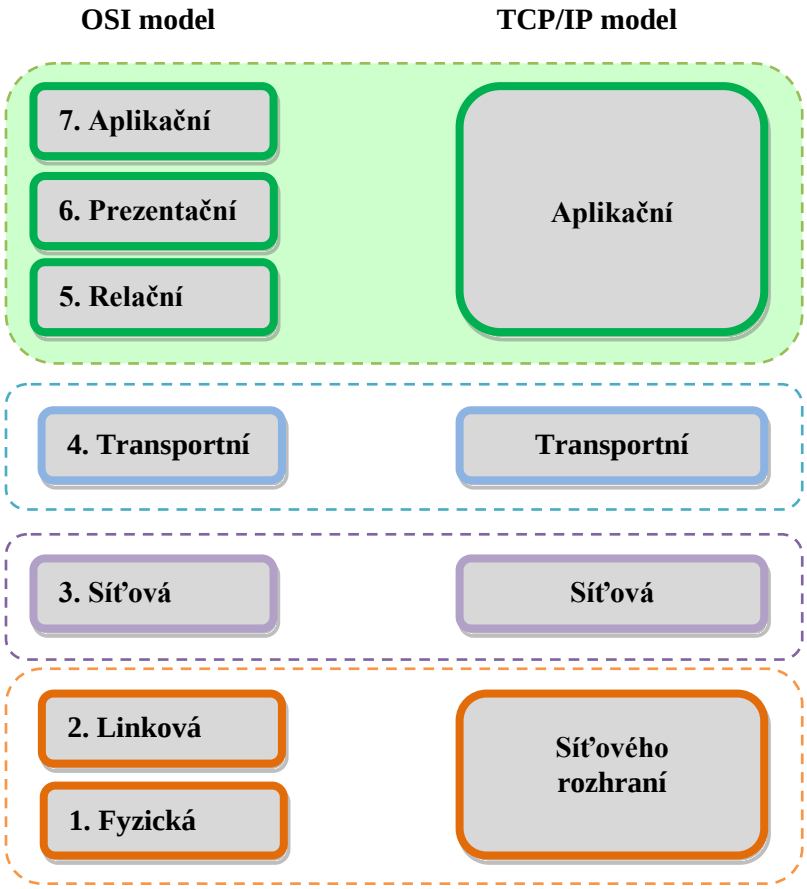
Obr. 8.3 TCP/IP model

8.4 Porovnání modelů ISO OSI a TCP/IP

Při srovnání obou modelů se dostaneme ke klíčovým faktorům jednotlivých vrstev sítě. Je zřejmé, že výrobci zařízení a navrhovatelé standardů protokolů musí dodržet funkčnost odpovídajících vrstev modelu. Proto tyto modely dáme vedle sebe a určíme tak které vrstvy si navzájem odpovídají. Aplikační vrstva modelu TCP/IP odpovídá horním třem vrstvám referenčního OSI modelu. Tato vrstva komunikuje přímo s transportní vrstvou, takže všechny prezentační a relační funkcionality si musí zajistit sama. Naopak prostřední vrstvy, již zmíněná transportní a dále síťová si v obou modelech odpovídají. Úkolem transportní vrstvy je udržovat přenos dat mezi koncovými účastníky a to v obou směrech, podle druhu komunikace zajistit jejich spolehlivost a další faktory, o které se nižší vrstvy nestarají. Síťová vrstva pak na základě adresování pomocí IP adres zajišťuje nejvhodnější cestu v síti, přes mnoho síťových zařízení. Nestará se tak o faktory spolehlivosti, potvrzování a redukce toku dat, které jsou úkolem vyšší – transportní vrstvy. Spodní vrstva modelu TCP/IP odpovídá dvěma vrstvám OSI modelu: linkové a fyzické. Tyto dvě spodní vrstvy jsou natolik provázané, že v modelu TCP/IP nemá opodstatnění je oddělovat. Spodní vrstva je tedy vrstvou síťového rozhraní a má na starost nejen přenést po komunikačním médiu příslušný datový signál ale pomocí linkových protokolů a adresování MAC adresami zajistit doručení uživatelských dat na cílové zařízení.

Pojem k zapamatování: TCP/IP model

Čtyřvrstvý TCP/IP model odpovídá sedmivrstvému referenčnímu modelu tak, že aplikační vrstva slučuje funkce aplikační, prezentační a relační referenčního modelu a vrstva síťového rozhraní slučuje fyzickou a linkovou vrstvu. Transportní a síťová vrstva si v obou modelech odpovídá.

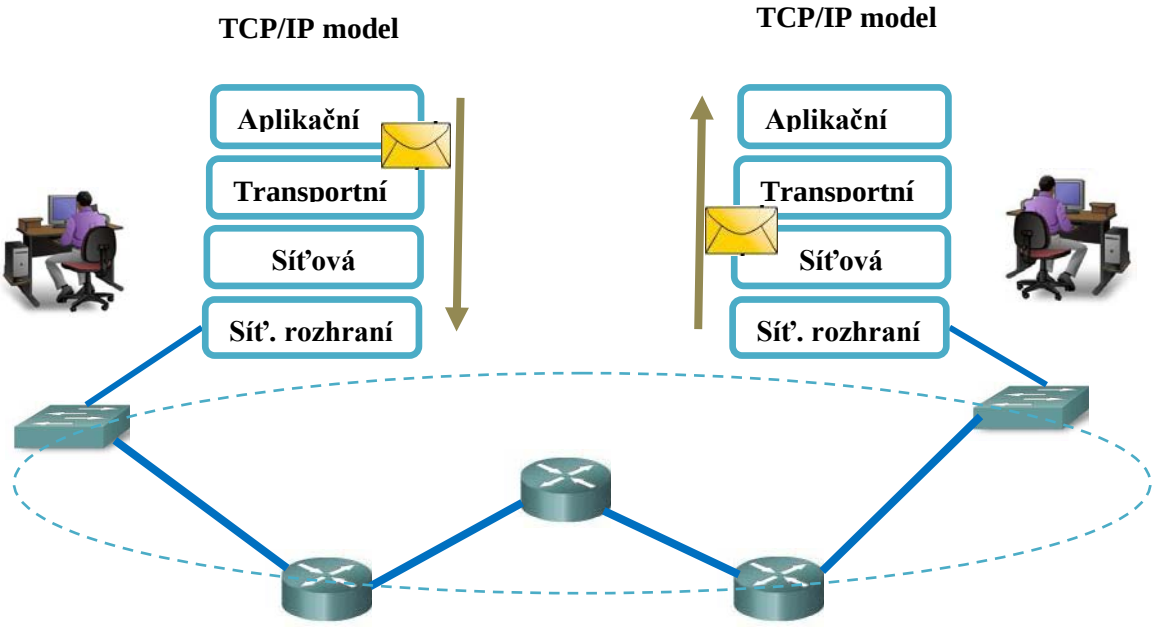


Obr. 8.4 Srovnání obou síťových modelů

8.5 Komunikační proces

TCP/IP model odpovídá protokolovému zásobníku, který definuje v podstatě komunikační proces. Na obou stranách konverzace je implementována rodina protokolů TCP/IP. Popíšeme si postup, kdy se jedno zařízení pokusí realizovat komunikační proces s druhým zařízením. Uvedený postup platí obecně pro popisované množství dat. Realita je taková, že uživatelská data jsou zpracována do paketů po určitých částech, podle maximální délky datové části paketu. Proto uvedený postup platí obecně pro pakety, kterých bude při vzájemné komunikace tisíce.

1. Vytvoří se uživatelská data na zdrojovém zařízení, například vyplněním webového formuláře nebo napsáním elektronické pošty.
2. Následuje tzv. proces segmentace, kdy se jednotlivá data rozdělí na informační celky, zároveň se provede zapouzdření neboli enkapsulace do paketů, které se na jednotlivých vrstvách TCP/IP modelu předávají nižším vrstvám.
3. Následuje generování impulsů na přenosovém médiu, které reprezentuje příslušná data.
4. Dále se transportují data po síti přes mnoho síťových zařízení s příslušným komunikačním médiem.
5. Nakonec data dorazí k cílovému zařízení, kde jsou přijata spodní vrstvou síťového rozhraní.
6. Následuje proces dekapsulace, neboli rozbalení paketu na jednotlivých vrstvách TCP/IP modelu.
7. Cílová data jsou předány aplikační vrstvě a konkrétní aplikaci, pro kterou jsou určena.



Obr. 8.5 Komunikační proces

Uvedený postup je samozřejmě velice zjednodušený. Ve skutečnosti každé síťové zařízení, provádí dekapsulaci obdrženého paketu, samozřejmě ne až do poslední vrstvy, protože ho cílová data nezajímají. Síťové zprostředkující zařízení ale bude dekapsulovat paket na úroveň své vrstvy a opět paket zabalí a pošle k následujícímu síťovému zařízení. Až poslední zařízení, které je cílové rozbaluje paket až po aplikační data.

Pojem k zapamatování: Komunikační proces

Komunikační proces mezi dvěma zařízeními probíhá tak, že ze zdrojového zařízení jsou data segmentována a zapouzdřena do informačních jednotek, které putují jednotlivými vrstvami protokolového zásobníku. Po přijetí koncovým zařízením jsou opět data na jednotlivých vrstvách rozbalována a samotná data předány aplikační vrstvě.

8.6 Proces zapouzdření a rozbalení paketu

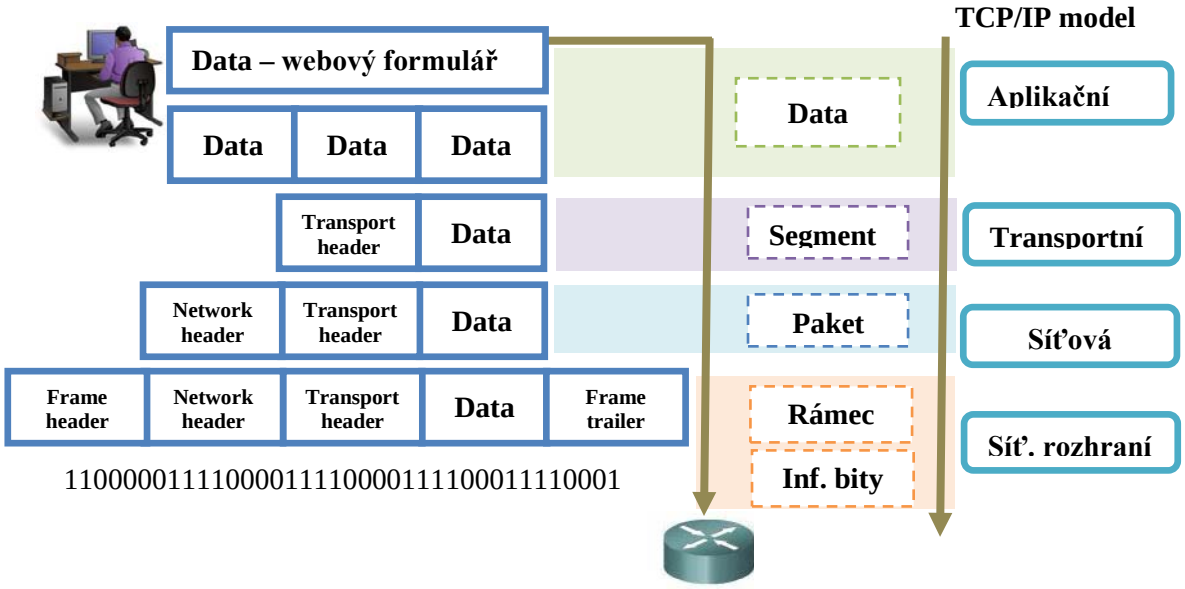
Aplikace předává své data do protokolového zásobníku napříč vrstvami směrem dolů, při tomto procesu se data postupně balí do určitých celků na každé vrstvě tak, aby byly na těchto vrstvách identifikovatelné v průběhu a na konci své cesty. Tomuto procesu se říká proces zapouzdření, neboli enkapsulace.

Pojem k zapamatování: PDU – protokolová datová jednotka

Jednotlivým informačním úsekům, pro které jsme doposud používali obecný termín paket, bychom měli říkat protokolová datová jednotka (*Packet Data Unit*). Tento termín je zaveden právě z důvodu zapouzdřovacího procesu. Tato datová jednotka (*PDU*) se na každé vrstvě jmenuje jinak a proto by bylo nesprávné používat termín paket, neboť to je již tvar PDU na síťové vrstvě.

Formát protokolové jednotky na jednotlivých vrstvách je následující:

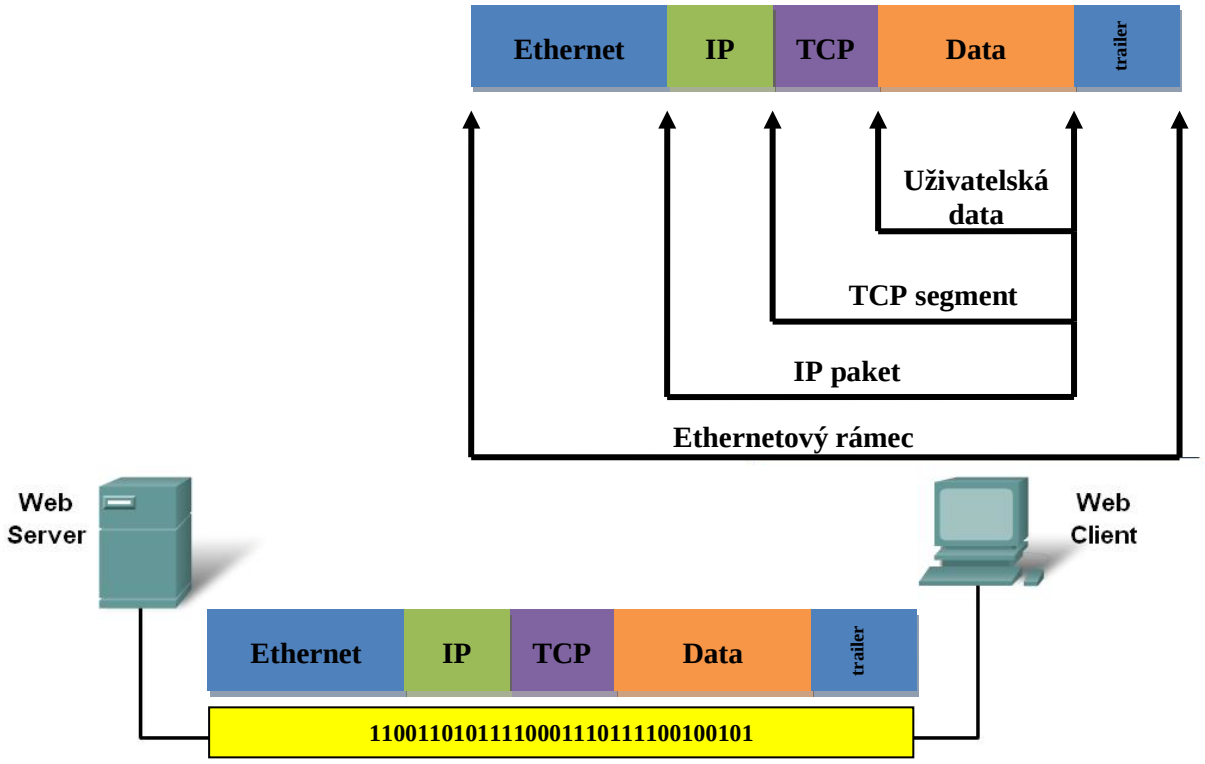
- Data – Na aplikační vrstvě je pro PDU používán termín data.
- Segment – Transportní vrstva přidá na začátek hlavičku svého konkrétního protokolu.
- Paket – na síťové vrstvě je k segmentu z předchozí vrstvy přidána na začátek hlavička síťového protokolu.
- Rámec – Na úrovni vrstvy síťového rozhraní protokol přibaluje nejen hlavičku ale také zakončení rámce.
- Bity – Když je rámec spodní vrstvou překódován na posloupnost bitů, je vše připraveno na vyslání po komunikačním médiu.



Obr. 8.6 Proces zapouzdření do informačního rámce

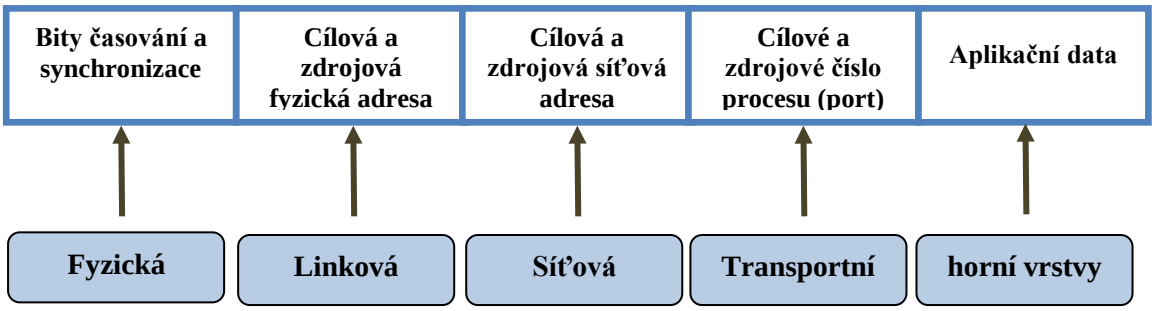
Pojem k zapamatování: Header, Trailer, Frame, Encapsulation, Decapsulation

Pojmy ***Header*** – hlavička a ***Trailer*** - zakončení jsou anglické názvy, které budeme používat v důsledku jejich kratšího zápisu. Rovněž lze někdy použít místo termínu rámec angl. Výraz ***Frame***, neboť tento termín již téměř zdomácněl. Pojem zapouzdření se anglicky nazývá enkapsulace (*encapsulation*) a naopak rozbalení - dekapsulace (*decapsulation*).



Obr. 8.7 Proces vyslání a obdržení zprávy

Na obrázku 8.7 vidíme process vyslání zprávy, konkrétně nějaké HTML stránky z web serveru do klientského počítače. Nejprve se musí vyslaná aplikační data rozdělit do jednotlivých segment a ty budou opatřeny hlavičkou TCP protokolu. V této hlavičce jsou udržované informace o tom, jaký aplikační proces na cílovém počítači si vyžádal zasílaná data. Transportní vrstva tyto segment vyšlou síťové vrstvě, která segment zabalí do paketu a opatří ho svou IP hlavičkou. IP hlavička obsahuje zdrojovou a cílovou IP adresu a také další nezbytné informace pro správné doručení do cílového místa. Nakonec je paket předán spodní vrstvě síťového rozhraní, kde je zabalen do Ethernetového rámce. Tento rámec obsahuje nejen hlavičku ale také zakončení, ve kterém je potřeba provést kontrolní součet, zdali rámec v cílovém místě dojde nepoškozen. V hlavičce rámce musí být opět adresa cílového zařízení, tentokrát je zde uvedena fyzická adresa zařízení. Na závěr je rámec kódován podle příslušného předpisu komunikačního standardu a vyslán na komunikační medium. Při obdržení zprávy je přijatá posloupnost signálů dekodována na data, je zkontrolován kontrolní součet rámce a v případě úspěchu jsou PDU na jednotlivých vrstvách rozbalována a předána vyšším vrstvám.

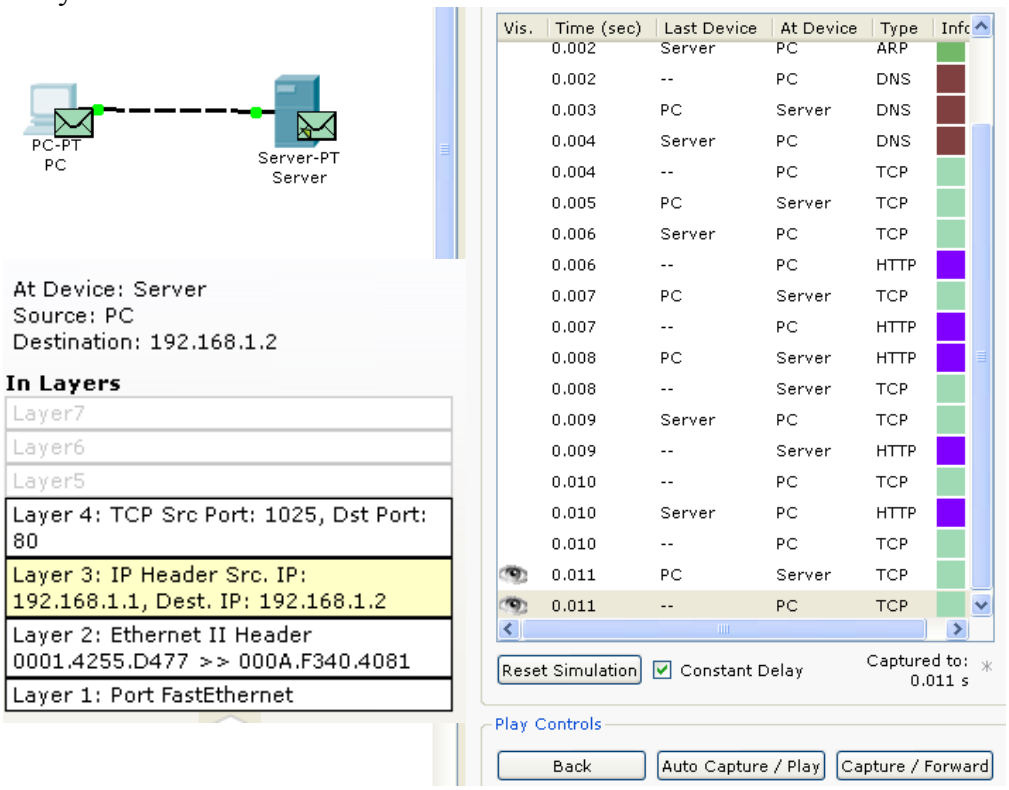


Obr. 8.8 Identifikace zařízení na vrstvách referenčního modelu

Na jednotlivých vrstvách se jedná o různé typy adresování, které musí korespondovat s cílovým zařízením. Na obrázku 8.8 vidíme, jak jsou jednotlivé typy adres zařazeny do vrstev referenčního modelu. Na linkové vrstvě se jedná o unikátní fyzickou neboli MAC adresu, na síťové vrstvě se jedná o IP adresu a na transportní vrstvě se jedná o jednoznačně definované číslo portu konkrétní aplikace. Jednotlivá adresování na všech vrstvách si probereme v dalších kapitolách.

Příklad 8.1 – simulace komunikace PC a web serveru v programu PT

Na obrázku vidíme přímo komunikaci PC a web serveru v programu Packet Tracer. V programu máme možnost spustit simulační mód a vidět tak průběh komunikace zařízení i s přeposíláním rámců, které si můžeme detailně prohlížet a vidět detaily na jednotlivých vrstvách síťového modelu.



Ke kapitole 8 je přiřazena demonstrační animace č. 5a č.6

Animace č. 5 obsahuje ukázku funkčnosti simulačního módu, po propojení dvou zařízení nastavení IP adresace a vytvoření jednoduchého komunikačního kanálu pro formát PING zprávy, ukázku simulace PING a detailní obsah paketu na jednotlivých vrstvách. Animace č.6 obsahuje ukázku komunikace mezi PC a webovým serverem při PINGU ale také při zobrazení webové stránky servru v počítači klienta, nastavení DNS serveru, konfiguraci home page, simulaci PINGu a sledování paketů při zapnutém filtru na protokol DNS a http.

Shrnutí kapitoly

Vzájemnou interakcí protokolů se vytváří tzv. protokolový zásobník (*Protocol stack*). Ten v sobě obsahuje jednotlivé vrstvy protokolů, hierarchicky právě od aplikačních dat až po zmíněné nuly a jedničky na nejnižší. Protokolový zásobník v sobě obsahuje 4 vrstvy TCP/IP modelu: aplikační, transportní, síťovou a vrstvu síťového rozhraní. Tyto 4 vrstvy zabalují a rozbalují na své cestě datový paket.

Při vytváření architektury síťového modelu se používá díky interakcí protokolů právě vícevrstvý model. Tento model definuje úkoly každé vrstvy. To umožňuje výrobcům

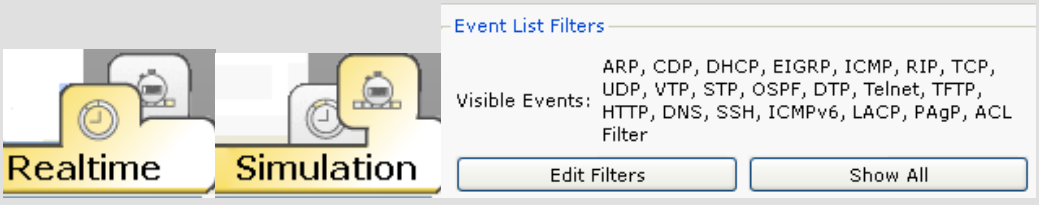
odlišných technologií vytvářet protokoly podle standardů tak, aby protokoly na jednotlivých vrstvách si dokázaly porozumět. Hlavním úkolem referenčního modelu je definice norem pro účely propojování počítačových sítí. Norma nespecifikuje realizaci systémů, ale uvádí všeobecné principy sedmivrstvé síťové architektury. Popisuje vrstvy, jejich funkce a služby, nikoliv samotné protokoly, jejichž implementace není úkolem síťového modelu. Čtyřvrstvý TCP/IP model odpovídá sedmivrstvému referenčnímu modelu tak, že aplikační vrstva slučuje funkce aplikační, prezentační a relační referenčního modelu a vrstva síťového rozhraní slučuje fyzickou a linkovou vrstvu. Transportní a síťová vrstva si v obou modelech odpovídá.

TCP/IP model odpovídá protokolovému zásobníku, který definuje v podstatě komunikační proces. Na obou stranách konverzace je implementována rodina protokolů TCP/IP. Nejprve zdrojový počítač vytvoří uživatelská data například vyplněním webového formuláře nebo napsáním elektronické pošty. Následuje tzv. proces segmentace, kdy se jednotlivá data rozdělí na informační celky, zároveň se provede zapouzdření neboli enkapsulace do paketů, které se na jednotlivých vrstvách TCP/IP modelu předávají nižším vrstvám. Následuje generování impulsů na přenosovém médiu, které reprezentuje příslušná data. Poté se transportují data po síti přes mnoho síťových zařízení s příslušným komunikačním médiem a nakonec data dorazí k cílovému zařízení, kde jsou přijata spodní vrstvou síťového rozhraní. Zde následuje proces dekapsulace, neboli rozbalení paketu na jednotlivých vrstvách TCP/IP modelu a cílová data jsou předány aplikační vrstvě a konkrétní aplikaci pro kterou jsou určena.
Formát protokolové jednotky na jednotlivých vrstvách je následující:

- Data – Na aplikační vrstvě je pro PDU používán termín data.
- Segment – Transportní vrstva přidá na začátek hlavičku svého konkrétního protokolu.
- Paket – na síťové vrstvě je k segmentu z předchozí vrstvy přidána na začátek hlavička síťového protokolu.
- Rámec – Na úrovni vrstvy síťového rozhraní protokol přibaluje nejen hlavičku ale také zakončení rámce.
- Bity – Když je rámec spodní vrstvou překódován na posloupnost bitů, je vše připraveno na vyslání po komunikačním médiu.

Úkol k řešení 8.1 – Simulační mód programu Packet Tracer 5.0

V aplikaci Packet Tracer se seznamte s přechodem do simulačního režimu pomocí přepínačů v pravém dolním rohu plochy programu.
V simulačním režimu si všimněte výběru typů protokolů, které lze v simulacích sledovat.



Úkol k řešení 8.2 – Simulace komunikace v programu Packet Tracer 5.0

Realizujte řešený příklad 8.1- komunikace PC a web servru. Příklad je na přiloženém CD i výukovém serveru v kapitole úkoly k řešení.

Kontrolní otázka 8.1

Co rozumíme pod pojmem vzájemná interakce protokolů?

Kontrolní otázka 8.2

Co je to *Protocol Stack*?

Kontrolní otázka 8.3

Proč zavádíme referenční ISO OSI model?

Kontrolní otázka 8.4

Které vrstvy referenčního modelu chybějí v modelu TCP/IP? Jak nahradíme jejich funkčnost?

Kontrolní otázka 8.5

Popište komunikační proces.

Kontrolní otázka 8.6

Co je to enkapsulace a dekapsulace paketu?

Kontrolní otázka 8.7

Popište formát paketu na jednotlivých vrstvách.

Kontrolní otázka 8.8

Co znamenají termíny *Header* a *Trailer*?

Kontrolní otázka 8.9

Co rozumíme pod pojmem PDU?

9. APLIKAČNÍ A TRANSPORTNÍ VRSTVA

Čas ke studiu: 3 hodiny

Cíl Po prostudování této kapitoly budete umět

- popsat úkoly aplikační vrstvy
- charakterizovat nejznámější aplikační protokoly
- popsat úkoly transportní vrstvy
- definovat protokoly TCP a UDP
- vysvětlit princip spolehlivé TCP komunikace
- vysvětlit podstatu komunikace prostřednictvím portů transportní vrstvy

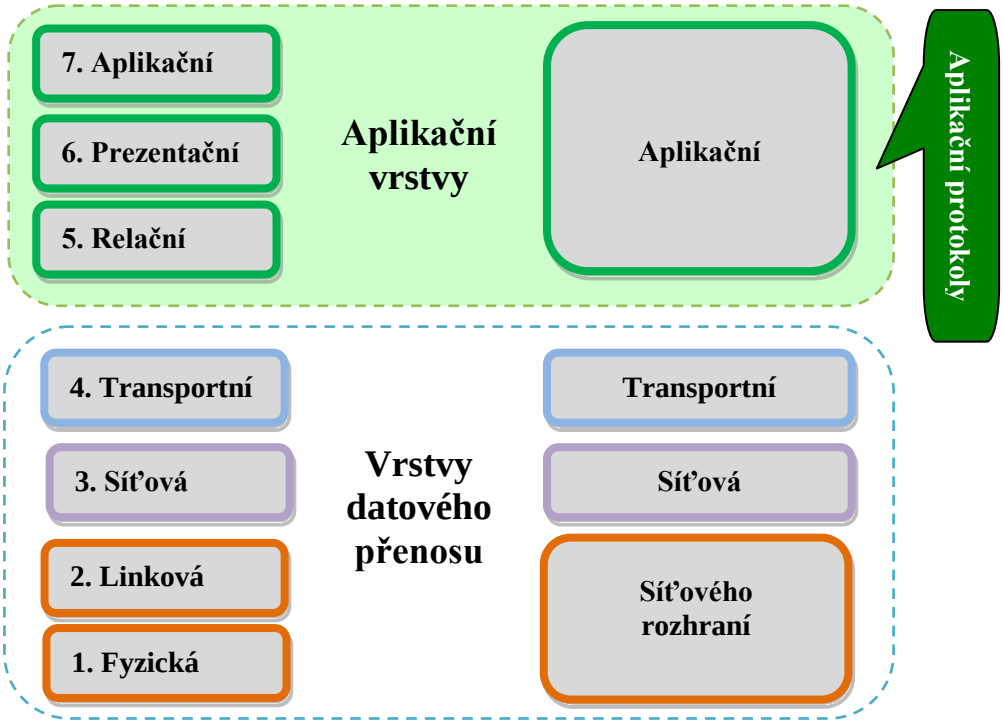
Výklad

Po představení obou síťových modelů následuje detailní popis jednotlivých vrstev včetně protokolů, které na těchto vrstvách operují. V této kapitole si představíme horní dvě vrstvy z pohledu modelu TCP/IP.

9.1 Úloha aplikační vrstvy

Aplikační vrstva je horní vrstvou obou síťových modelů, její úlohou je zajistit rozhraní mezi aplikace mi spuštěnými na jednotlivých zařízeních, které potřebují komunikovat po počítačové síti a nižšími vrstvami, které zprostředkovávají síťové a přenosové funkce.
Hlavní rozdíl mezi OSI a TCP/IP aplikační vrstvou je ten, že aplikační vrstva modelu TCP/IP v sobě zahrnuje funkcionalitu všech tří horních vrstev OSI modelu. To znamená pro výrobce aplikačního softwaru a protokolů, že se musejí postarat o prezentaci a relaci vyměněných dat.
Prezentační vrstva OSI modelu definuje úlohu vlastního kódování a konverze aplikačních dat. Typickým příkladem prezentace dat jsou různé formáty kódových stránek textového formátu dat. Naproti tomu příkladem komprese dat jsou zvukové a obrazové formáty dat, např. MPEG, JPG, atd. Dá se říci, že úkoly prezentační vrstvy sin a sebe vzaly aplikace, typicky webové prohlížeče v podobě tzv. *Pluginů.*
Úkolem relační vrstvy je implementovat funkce udržování dialogu mezi zdrojovou a cílovou aplikací. Podstatou je udržovat dialog, v případě chyb jej obnovit či ukončit. I tuto úlohu si vzali na starost aplikace nejvyšší vrstvy. Zůstaňme u příkladu webového prohlížeče. Zde jistě znáte pojem *cookies* jako pomůcku dočasně udržované relace, pokročilejší serverové skriptovací jazyky implementují do aplikací tzv. *Session* proměnné, udržující relaci.
Mezi nejznámější aplikační protokoly rodiny TCP/IP protokolů patří protokoly zajišťující výměnu uživatelských dat a nezbytných informací pro chod aplikací využívajících internet Mezi tyto protokoly patří:

- Hypertext Transfer Protocol (HTTP) pro přenos dat do webových prohlížečů
- Domain Name Service Protocol (DNS) pro překlad známých Internetových adres na IP adresy.
- Simple Mail Transfer Protocol (SMTP) pro odesílání elektronické pošty.
- Telnet protokol pro vzdálený konzolový přístup na vzdálené zařízení.
- File Transfer Protocol (FTP) pro přenos souborů mezi koncovými zařízení.



Obr. 9.1 Aplikační vrstvy síťového modelu

Pojem k zapamatování: Protokol aplikační vrstvy

Hlavním úkolem protokolu aplikační vrstvy je výměna dat mezi aplikacemi běžícími na zdrojovém a cílovém zařízení komunikace.

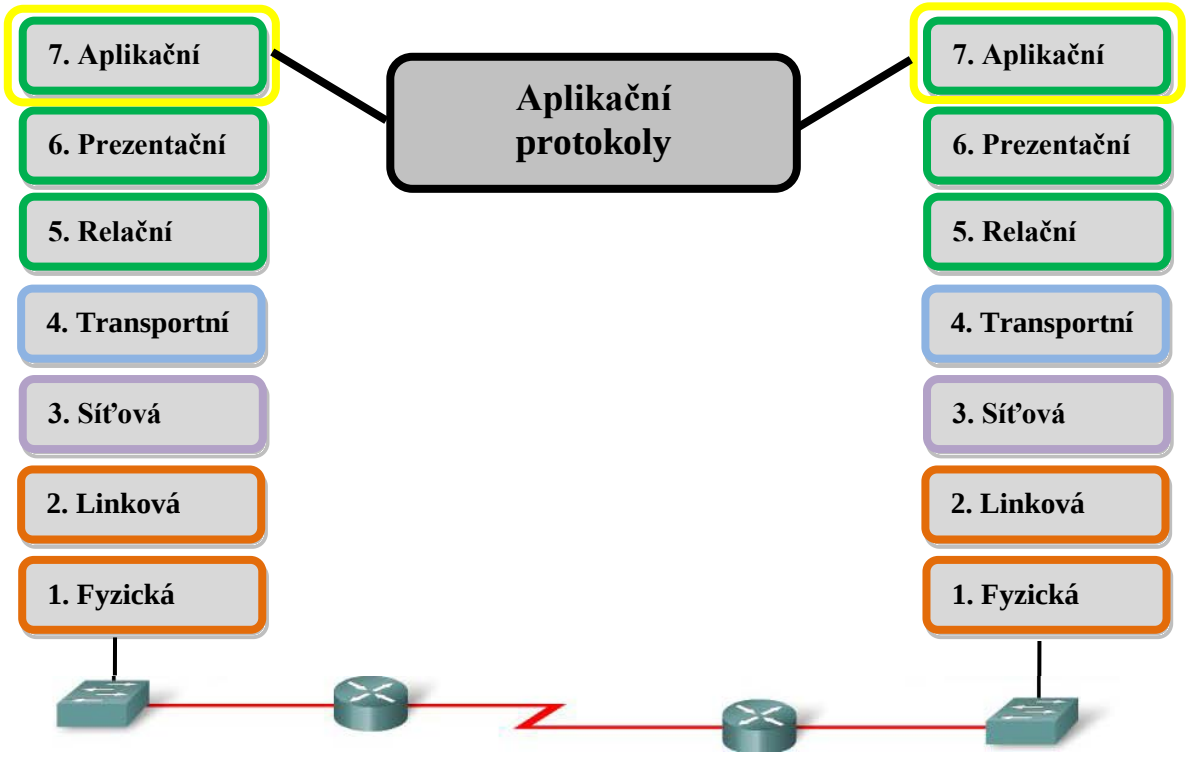
Aplikačních protokolů je samozřejmě daleko vice a jejich rodina se neustále rozrůstá s rozvojem nových internetových a dalších síťových aplikací.

9.2 Funkce aplikačních protokolů

Aplikační protokoly zajišťují komunikaci na obou stranách konverzace během dané relace. Jinými slovy, aby byla komunikace úspěšná, aplikační protokol musí být k dispozici na obou stranách konverzace. V úvodu výkladu o základních elementech počítačových sítí jsme zmiňovali fakt, že protokoly ustavují základní pravidla pro komunikaci koncových zařízení. To konkrétně znamená ustanovit základní tvar odpovědi na požadavek, definici datového formátu, mechanismus vyslání a přijetí dat, jejich potvrzení, popřípadě definice chybového stavu komunikace.
V současné a také budoucí době bude využívat síťovou komunikaci mnoho různých nových aplikací a aplikační protokoly společně s aplikacemi musí být schopny zajistit všechny funkce horních tří vrstev ISO modelu. Každý protokol má svůj specifický význam a charakterizuje požadavky konkrétní aplikace, pro kterou určen. Hlavní úkoly shrneme do následujícího pojmu k zapamatování:

Pojem k zapamatování: Funkce aplikačních protokolů

- definuje procesy na obou koncích komunikačního řetězce
- definuje typy vyslaných zpráv
- definuje syntaxi zpráv
- definuje způsob odesílání dat a podobu očekávané odpovědi
- definuje způsob spolupráce s nižší vrstvou síťového modelu

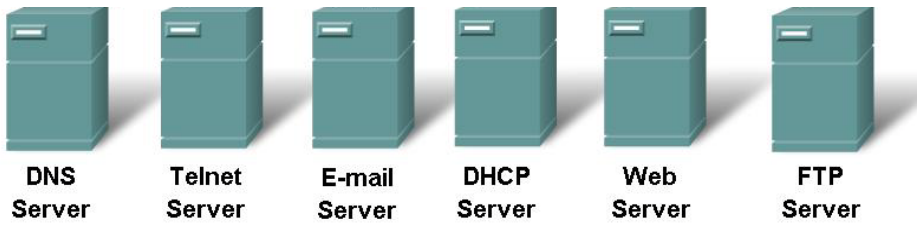


Obr. 9.2 Aplikační protokoly komunikují na obou stranách konverzace

Pojem k zapamatování: klient/server model

V modelu klient/server se zařízení, které se dožaduje informace, se nazývá klient a zařízení, které na požadavky odpovídá, se nazývá server. Tento model komunikace je navazován právě v aplikační vrstvě. Klient zahajuje komunikaci vysláním požadavku, na který server odpovídá. Protokoly aplikační vrstvy popisují formát požadavků a odpovědí mezi klientem a serverem.

Na obrázku 9.3 je znázorněno šest aplikačních serverů, na které nejčastěji uživatelé v roli klientů posílají své požadavky. Použití některých aplikačních serverů aplikacemi jako webový prohlížeč či poštovní klient jsou pro uživatelé jednoznačně pochopitelné, avšak o některých z nich uživatelé vůbec neví, přesto, že by se bez nich vůbec neobešli a používají je v souvislosti s prací v síti. Tím je míněno např. použití DHCP serveru pro přidělení IP adresy, či DNS serveru, bez kterého by uživatel musel místo www adresy psát do prohlížeče pro něj nezapamatovatelnou IP adresu.

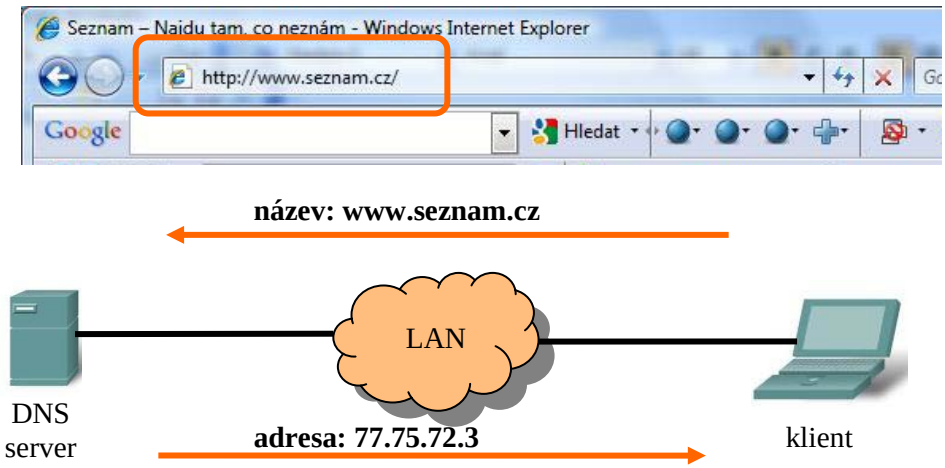


Obr. 9.3 Aplikační servery

9.3 Aplikační protokoly a jejich služby

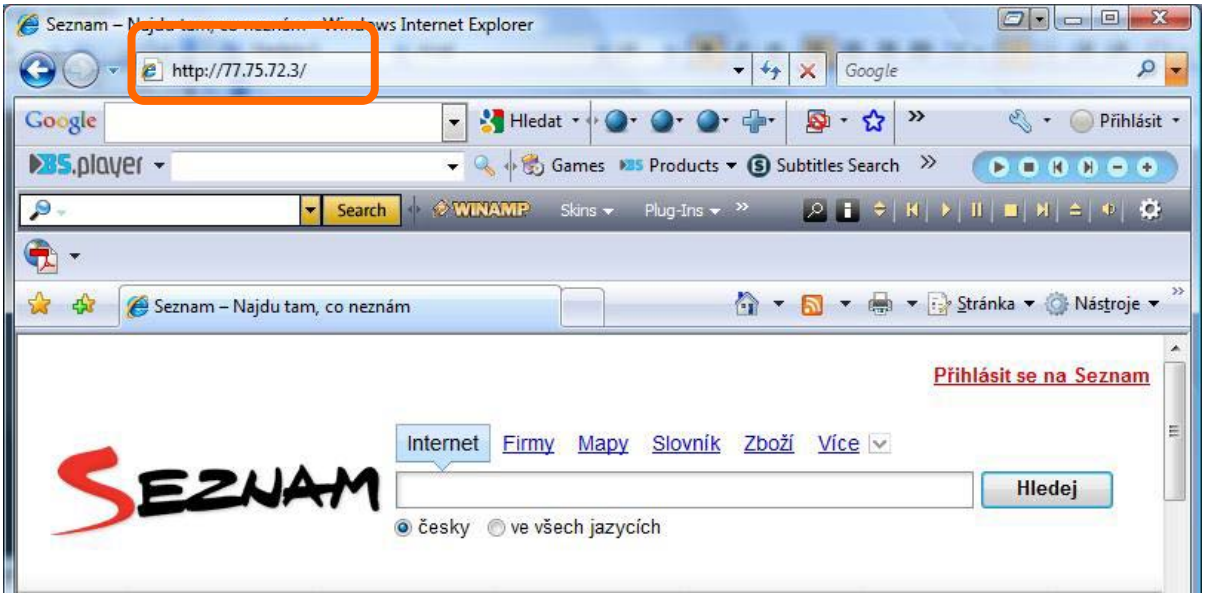
Protokol DNS – Domain Name System

Tak jak lidé komunikují prostřednictvím svého jazyka, kde postatou komunikace jsou zapamatovatelná hesla a názvy, tak počítače komunikují prostřednictvím nul a jedniček. IP adresa je také reprezentována v PC v binární formě, ale protože s ní člověk mnohdy pracuje, převedl si ji na decimální srozumitelný formát. Avšak i pamatování stovek a více IP adres je pro člověka neřešitelnou úlohou atak máme v mnoha případech k dispozici zástupné názvy, tzv. *aliasy*. Podobným případem je i problematika názvů webových domén a adres. Protože pro člověka tyto jména dávají smysl, je webová adresa v lidském srozumitelném formátu. Avšak počítač potřebuje jednoznačný údaj o adrese webového serveru a tak potřebujeme prostředníka, který převede jmenný název na IP adresu.



Obr. 9.4 DNS server převádí jmenný adresní název na IP adresu

Tímto prostředníkem se stává DNS server, tzv. doménový jmenný systémový server. Celý systém názvu webových adres je založen na komunikaci doménových serverů, které mají odkazy na další názvy domén. Jistě jste někdy slyšeli o doménách prvního řádu. Jsou jimi národní domény (např. .cz .sk), neomezené domény (.org, .com, .net, .info), speciální domény (.edu, .gov, .biz) a další. Pod těmito doménami se nacházejí domény druhé úrovně, atd.

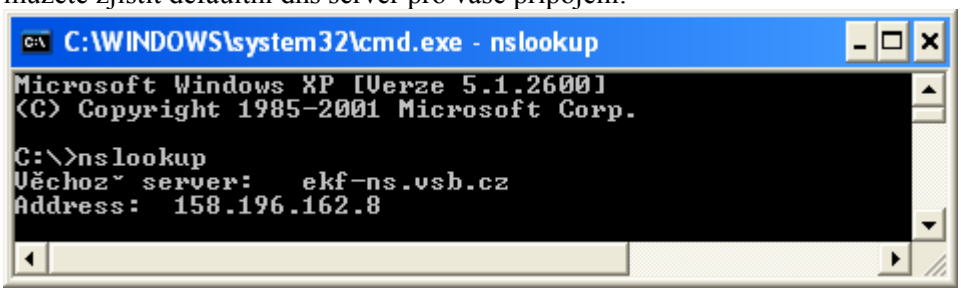


Obr. 9.5 Bez přítomnosti DNS serveru bychom museli v prohlížeči zadávat IP adresy

Jmenné servery jsou autoritativní, to znamená, že mají informace o názvech serverů v příslušných doménách. Poslední autoritativní jmenný server poskytne IP adresu pro přeložení ze jména uloženého v URL adrese.

Korespondenční úkol 9.1 – utilita nslookup

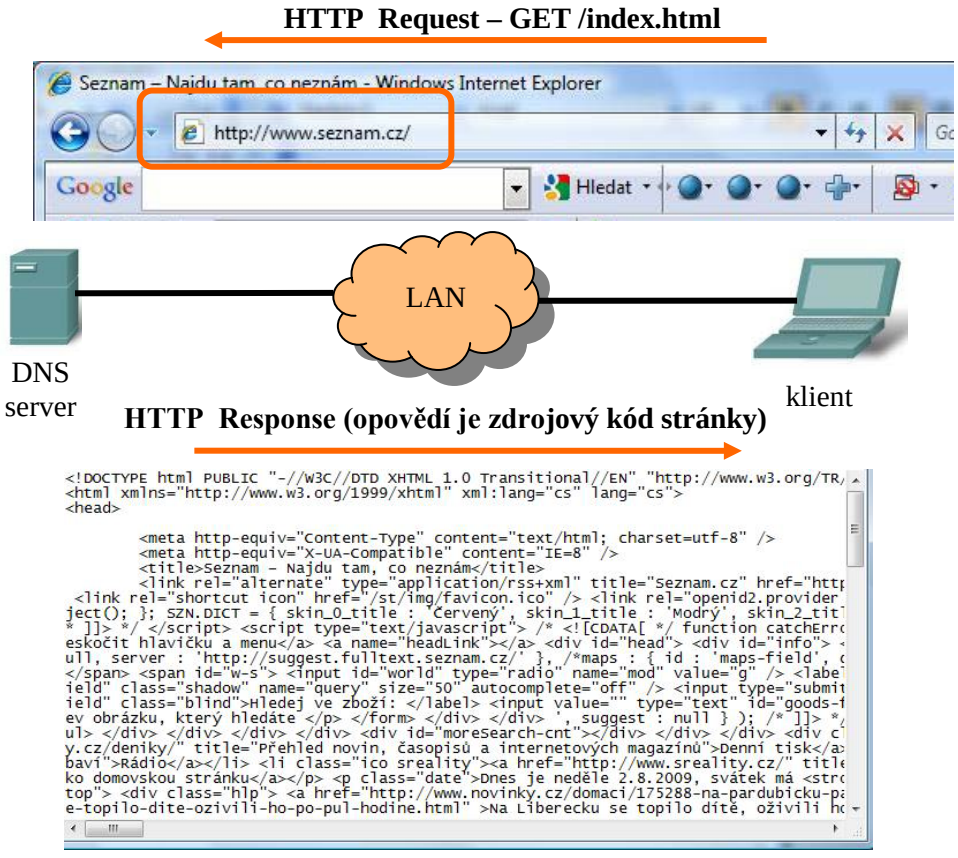
Operační systém vašeho PC je vybaven programem *nslookup*. Pomocí tohoto programu můžete zjistit defaultní dns server pro vaše připojení.



Otevřete příkazový řádek neboli *Command prompt* a napište do něj příkaz *nslookup*. Zkontrolujte tak přítomnost DNS serveru.

Protokol HTTP – Web server

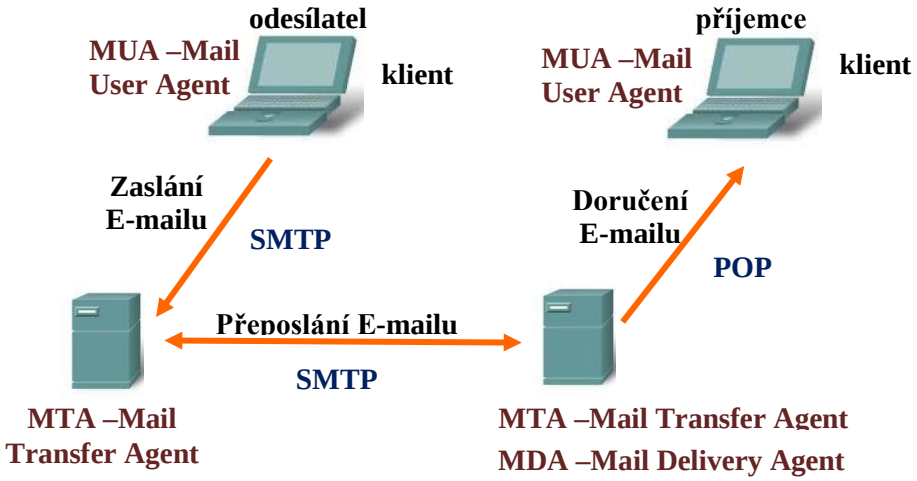
Rovněž komunikace prostřednictvím http protokolu funguje na principu modelu klient/server. Klient vyšle na webový server prostřednictvím protokolu požadavek a webový server na požadavek zareaguje odpovědí. Běžnými požadavky klienta jsou příkazy GET, POST a PUT. Požadavek GET je formou žádosti klienta o poskytnutí dat. Požadavky POST a PUT realizují vyslání *uploadu* dat na webový server.



Obr. 9.6 Http komunikace klient/server

Poštovní server – protokoly POP a SMTP

Elektronická pošta je jednou z nejoblíbenějších a nejčastěji používaných služeb počítačových sítí. Celý proces posílání pošty se opět řídí modelem klient/server s tím, že klient je v roli jak odesílatele pomocí protokolu SMTP (*Simple Mail Transfer protocol*), tak v roli příjemce, kdy poštu přijímá pomocí protokolu POP (*Post office Protocol*) nebo IMAP(*Internet Message Access Protocol*). Klientovi se říká ***Mail User Agent*** (*MUA*). Poštovní server se účastní dvou rolí, buďto má klienta ve svém seznamu příjemců a zprávu doručí, v tomto případě je jeho role ***Mail Delivery Agent*** (*MDA*) anebo zprávu přepošle dalšímu emailovému serveru. V tomto případě je jeho role ***Mail Transfer Agent*** (*MTA*).



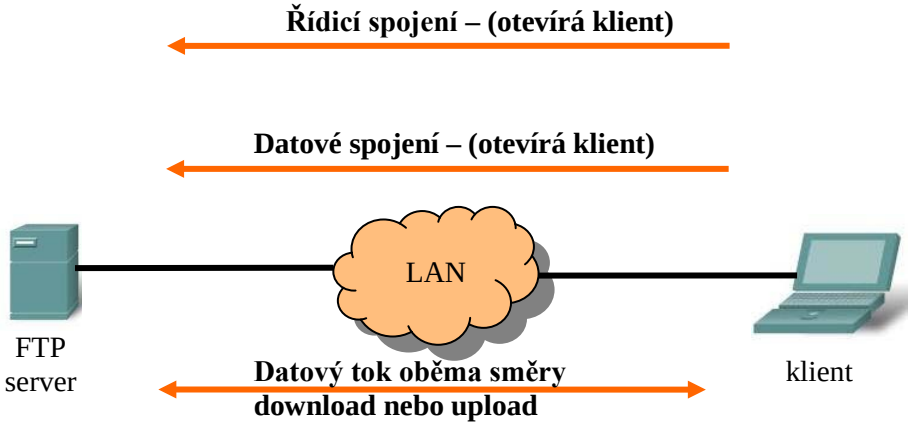
Obr. 9.7 Proces zaslání a příjmu pošty pomocí MUA, MTA a MDA

Korespondenční úkol 9.2 – protokoly pro příjem pošty

Pravděpodobně i vy používáte nějakého e-mailového klienta. Používáte pro příjem pošty protokol POP3 nebo IMAP4 ? Zjistěte, jaký je rozdíl v použití těchto protokolů.

Protokol FTP – File Transfer Protocol

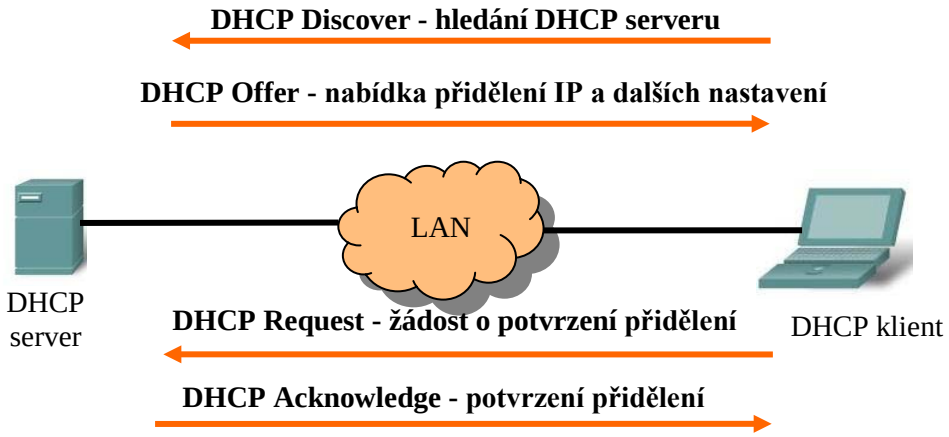
Jistě jste někdy potřebovali někomu poslat velké množství souborů a využít elektronickou poštu pro malou kapacitu poštovních schránek nebylo možné. Pak jste určitě použili nějaký program, který pomocí FTP protokolu povolí soubory odeslat (*upload*) nebo naopak přijmout (*download*). V jednu chvíli však lze pomocí protokolu pouze přijímat nebo naopak odesílat. Protokol FTP se liší od ostatních tím, že potřebuje pro svou činnost dvě spojení. Jedno spojení určuje řízení komunikace a druhé spojení je pro samotná data.



Obr. 9.8 Komunikace FTP klienta a serveru pomocí dvou otevřených spojení

Protokol DHCP – Dynamic Host Configuration Protocol

Každý počítač v síti potřebuje být identifikován IP adresou. Pokud administrujete malou síť, můžete IP adresy nastavovat manuálně. Ale i v tomto případě se můžete dostat do různých problémů, jejichž řešení vám bude ubírat čas. Ve velkých sítích manuální konfigurace IP adres již nepřichází v úvahu vůbec. IP adresy jsou přidělovány tzv. DHCP serverem, který má k dispozici určité množství IP adres a je připraven je přidělit počítačům, které se připojí do sítě, v níž má DHCP server oblast své působnosti. IP adresu počítač nedostane navždy. DHCP server zapůjčuje IP adresy počítačům na určitou dobu a pokud je počítač odpojen ze sítě, IP adresa se vrátí zpět do množiny volných IP adres.



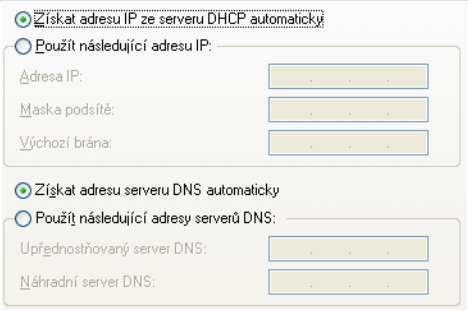
Obr. 9.9 DHCP komunikace

Proces přidělení IP adresy řídí komunikaci mezi PC v roli klienta a DHCP serveru. Jsou celkem zapotřebí čtyři komunikační fáze, než počítač obdrží IP adresu. Po připojení počítače do sítě počítač vyšle *broadcastový* paket tzv. DHP Discover, který hledá přítomnost DHCP serveru. *Broadcastový* znamená fakt, že je vyslán všem účastníkům sítě. Počítač totiž neví komu má žádost poslat, tak nejjednodušší je poslat paket všem. Tento způsob rozeslání žádosti není v počítačové síti neobvyklý, používá jej mnoho protokolů na různých vrstvách. *Broadcast* paket dojde jednotlivým zařízením a zařízení kterému je určen ho zpracuje, ostatní zařízení paket zahodí. *Discover* Paket tedy dojde na DHCP Server a ten odpoví paketem *DHCP Offer*, který obsahuje nabídku IP adresy, masky sítě, nastavení DNS servru a výchozí brány. Klient dalším požadavkem oznamuje, že s přidělenými údaji souhlasí a na závěr přijde ze serveru potvrzení. Tohle je samozřejmě ideální situace. Pokud se klient a server na přidělených údajích nedomluví, nebo přijde ze serveru zamítnutí (místo **ACK** přijde paket **NACK**), celý proces se musí opakovat.

Korespondenční úkol 9.3 - nastavení DHCP a DNS v počítači

Zkontrolujte si, zdali máte v nastavení síťových připojení ve vlastnostech TCP/IP protokolu nastaveno automatické přidělení IP adresy protokolem DHCP a přidělení IP adresy DNS serveru, či jej zadáváte ručně.

(*Nastavení – Síťová připojení – připojení k místní síti – vlastnosti)* . Pak volba *Protokol sítě Internet (TCP/IP).*



Telnet – terminál vzdáleného připojení

Telnet neboli virtuální terminál (*VTY*) je prostředek k tomu, abychom mohli na dálku spravovat konfigurace různých zařízení. Administrátor tak nemusí cestovat k zařízením, které má na starost, ale ze svého počítače spustí telnet klienta a spravuje v příkazovém řádku vzdálený počítač. Podmínkou je

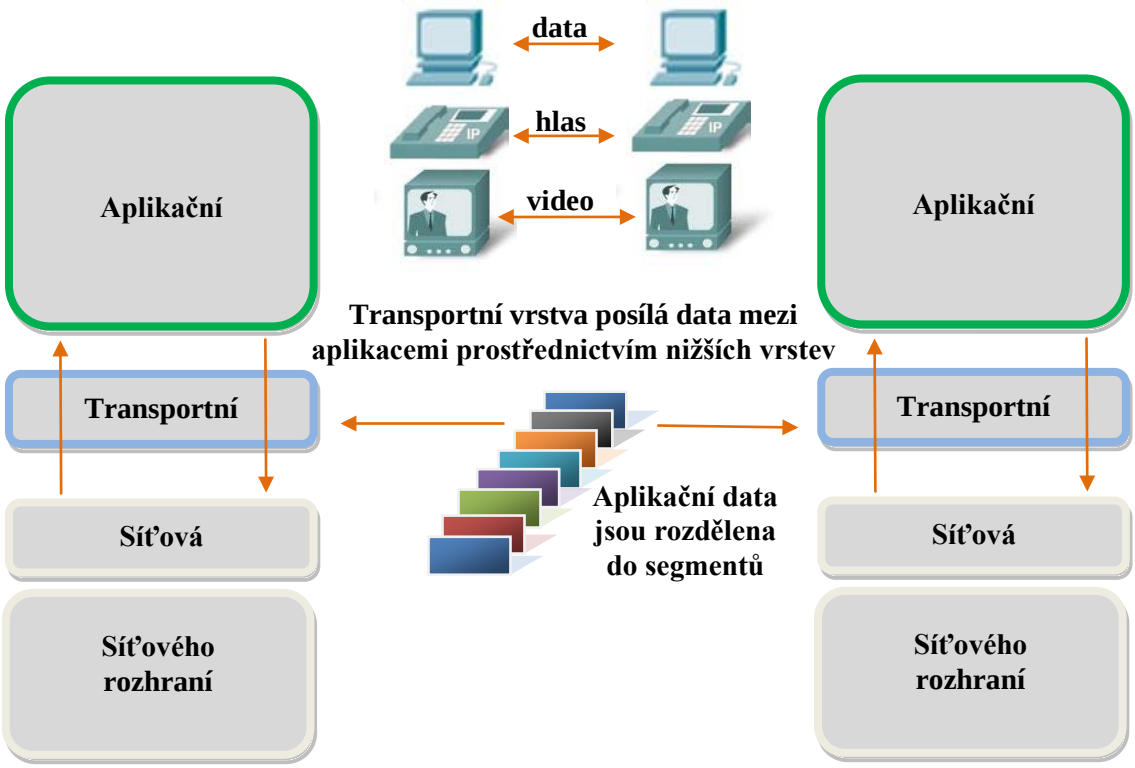
samozřejmě spuštěný telnet server, kterému se říká *daemon* na vzdáleném zařízení. Klient musí nejprve spojení se vzdáleným zařízením navázat, poté na zařízení pracuje a nakonec je vhodné spojení ukončit. V dnešní době je stále *telnet* velmi rozšířen, jeho nevýhodou však je, že veškeré komunikace probíhá v počítačové síti v textovém režimu včetně citlivých přihlašovacích údajů. Proto mnoho společností využívá pro vzdálenou správu obdobu telnetu tzv. SSH službu, která běží na obdobném principu avšak šifrovaně.

Obr. 9.10 Telnet funguje jako terminál ke vzdálenému zařízení

9.4 Úloha transportní vrstvy

Transportní vrstva zajišťuje segmentaci dat a jejich řízení nezbytně nutné pro různé typy komunikačních proudů. Její hlavní úlohy jsou:

- Řízení a sledování komunikace mezi aplikacemi na obou koncích.
- Segmentace dat a řízení každého segmentu
- Složení aplikačních dat zpět z proudů segmentů
- Identifikace různých typů aplikací



Obr. 9.11 Srovnání obou síťových modelů

Řízení a sledování komunikace mezi aplikacemi na obou koncích

Každý host může komunikovat prostřednictvím mnoha aplikací s okolním světem. Každá aplikace může navíc komunikovat s jedním či více vzdálenými hosty. Za řízení a sledování této komunikace je právě zodpovědná transportní vrstva.

Segmentace dat a řízení každého segmentu

Každá aplikace vytváří proud dat. Tyto data jsou dále rozkouskovány na segmenty a každá taková porce dat je opatřena hlavičkou transportní vrstvy.

Složení aplikačních dat zpět z proudů segmentů

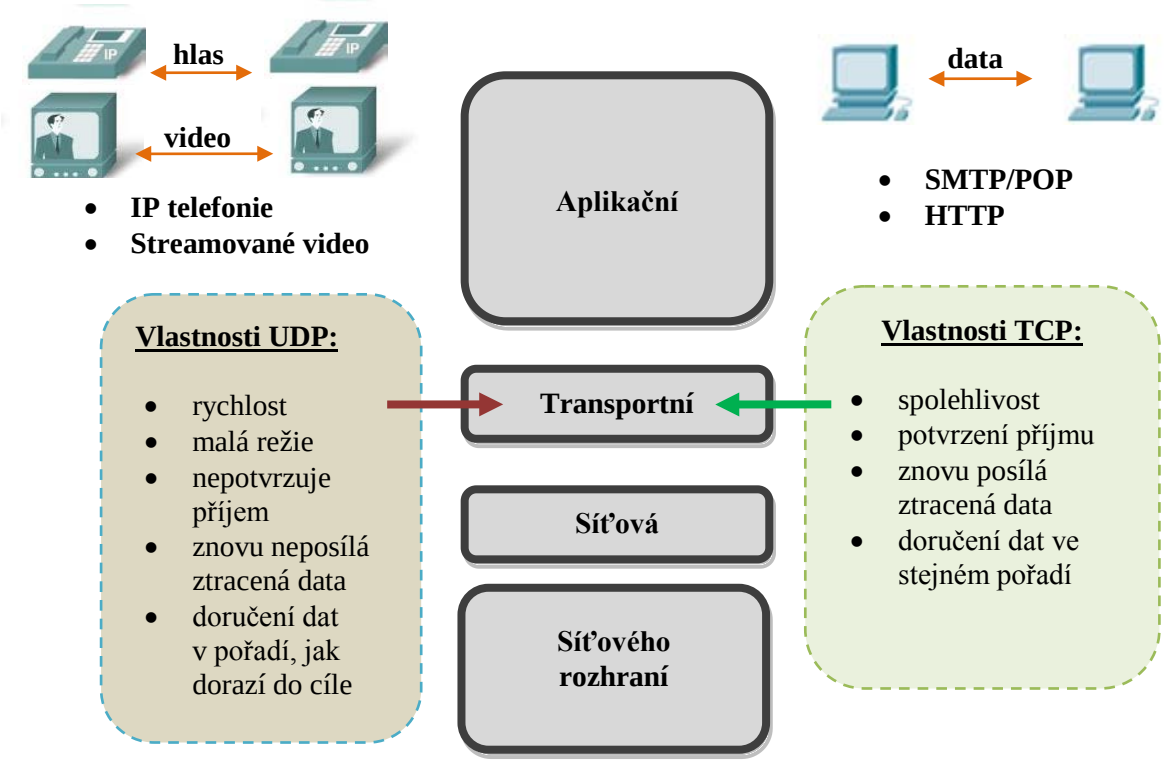
Přijímající strana obdrží data a provede proces dekapsulace. Úkolem transportní vrstvy zde je předat kompletní proud dat aplikační vrstvě. K tomu jsou využívány služby protokolů TCP nebo UDP, které se liší svoji povahou viz. následující kapitola.

Identifikace různých typů aplikací

Transportní vrstva musí vědět, které aplikaci má předat data. Musí ji tedy identifikovat. To se provádí mechanismem přiřazování portů. Port můžeme chápat jako zásuvku, do které bude transportní vrstva plnit data. Každá aplikace má svoji zásuvku a transportní vrstva ji identifikuje podle čísel.

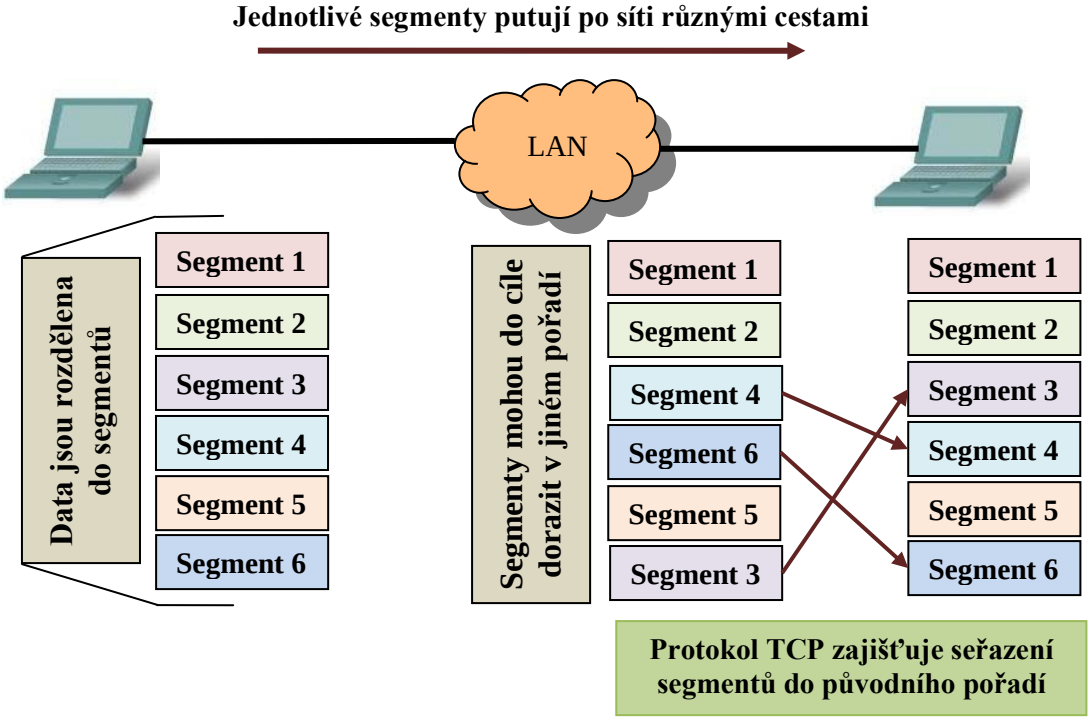
9.5 TCP a UDP

TCP a UDP protokoly jsou hlavními protokoly transportní vrstvy. Jejich úloha je totožná, je to řízení komunikace mezi aplikacemi. Rozdíl je však ve funkčnosti těchto protokolů. UDP je jednodušší protokol, který používá méně složitou strukturu PDU nazvanou datagramy. Datagramy jsou oproti segmentům TCP kratší, protože neobsahují sekvenční čísla a pole bytů pro potvrzení o přijetí paketu. Protokol UDP se používá v komunikaci mezi aplikacemi, které vyžadují rychlý přenos, který může být nespolehlivý. Proto není každý segment potvrzován. V praxi to znamená, že pokud konzumujete proud dat, např. Streamované video či hlasové služby po internet a nějaký datagram nedorazí, komunikace pokračuje dále, nemá vůbec žádný smysl se pokoušet o znovuzaslání nepřijatých dat. Naopak v aplikacích, které potřebují spolehlivost je využíván protokol TCP, který zajišťuje potvrzení příjmu segment a ztracená data posílá znovu.



Obr. 9.12 Srovnání obou síťových modelů

V praxi putují pakety od zdrojového počítače k cílovému zařízení po různých trasách, je obvyklé že pakety dorazí v jiném pořadí, než byly vyslány, tudíž po enkapsulaci do transportní vrstvy jsou v jiném pořadí I segmenty dat. Protokol TCP na rozdíl od protokolu UDP tyto segmenty uspořádá do správného pořadí.



Obr. 9.13 Seřazení segmentů do správného pořadí mechanismem protokolu TCP

9.6 Adresování portů

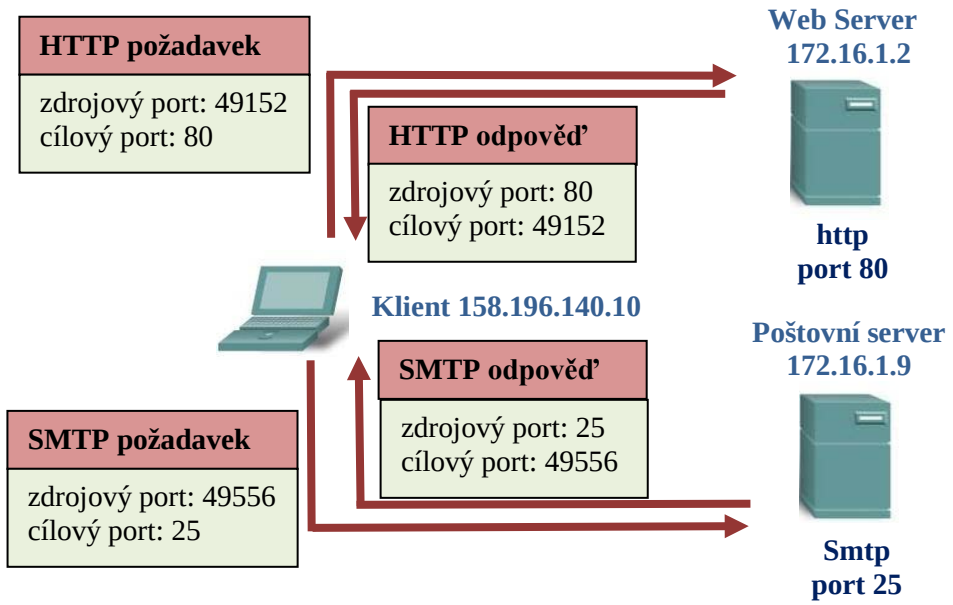
Adresování pomocí portů je jedním z nejdůležitějších poslání transportní vrstvy. TCP a UDP protokoly zajišťují komunikační přenos segmentů a datagramů mezi různými aplikacemi. Právě čísla portů jednoznačně identifikují komunikující strany. Zdrojový port je asociován s aplikací na straně klienta neboli lokálního uživatele, cílové číslo portu je asociováno s aplikací na straně hosta nebo serveru na vzdálené straně. Porty jsou přiřazovány různými způsoby v závislosti na tom, zdali se jedná o požadavek či odpověď. Zatímco server asociuje statické porty, klient dynamický přiděluje portům čísla pro každou konverzaci.

Rozsah portů	Skupina portů
0 - 1023	(dobře) známé porty
1024 - 49151	Registrované porty
49152 - 65535	Soukromé a dynamické porty

Obr 9.14 Rozdělení čísel portů

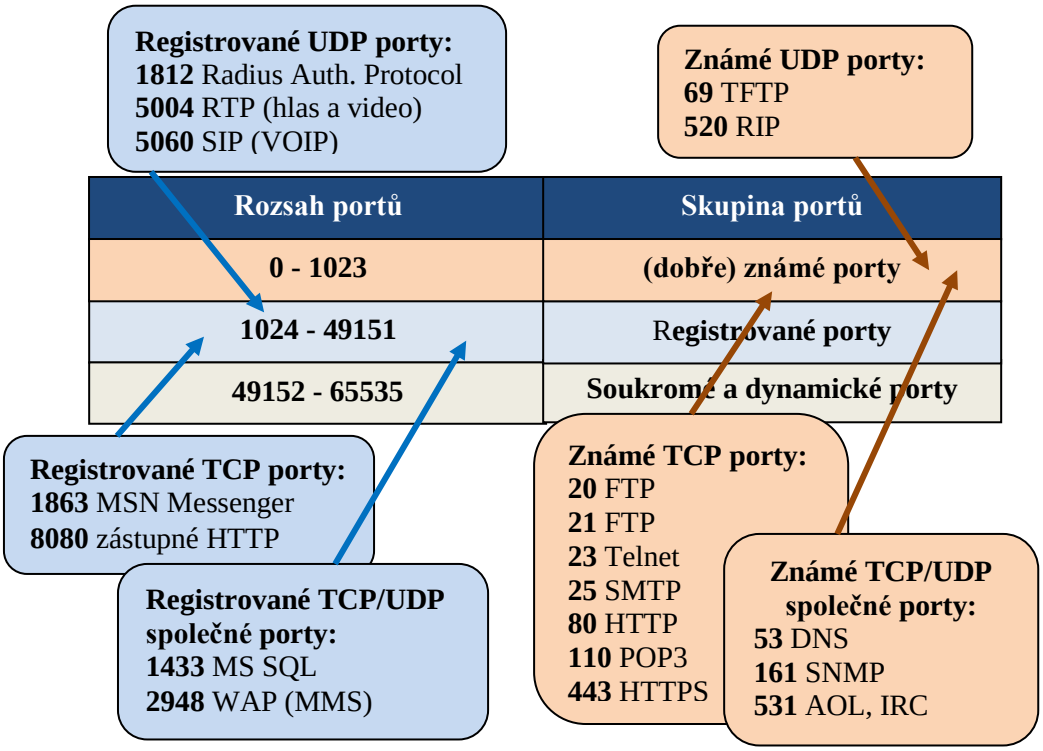
Na obrázku 9.14 vidíme rozdělení portů do třech skupin. Tzv. dobře známé porty (*well known ports*) jsou čísla přidělovaná službám nebo aplikacím, které dobře známe jako např. poštovní služby či HTTP. Registrované porty jsou přiřazeny uživatelským procesům nebo aplikacím a dynamické nebo soukromé porty jsou klientům přiřazeny, když inicializují komunikaci. Na obrázku 9.15 vidíme ukázku http a smtp komunikace. Klientský počítač vyslal na web server a poštovní server požadavek, inicializoval tedy komunikaci s dynamicky přiděleným portem. Např. klientský požadavek z IP adresy 158.196.140.10 bude opatřen zdrojovým portem 49152, resp. 49956 u SMTP komunikace. Výsledná

komunikace tak bude u http: 158.196.140.10:49152 → 172.16.1.2:80 a u smtp: 158.196.140.10:49556 → 172.16.1.9:25. Odpovědi pak budou zasílány zpět na odpovídající sokety - u http: 172.16.1.2:80 → 158.196.140.10:49152 a u smtp: 172.16.1.9:25 → 158.196.140.10:49556.



Obr. 9.15 Komunikace prostřednictvím portů transportní vrstvy

Některé aplikace využívají jak TCP segment tak UDP datagramů. Například DNS při obsloužení mnoha požadavků najednou odpovídá klientům pomocí UDP, naopak pokud je odpověď vyžadována s větší spolehlivostí, je voleno TCP. V obou případech je však použit stejný port, v případě DNS je to 53.



Obr 9.16 Asociace TCP, UDP a společných portů s aplikacemi

Ke kapitole 9 je přiřazena demonstrační animace č. 7

Animace č. 7 obsahuje ukázku komunikace mezi PC a webovým serverem v simulačním módu při prohlížení detailů paketů TCP a UDP. Při prohlížení struktury paketů je kladen důraz na velmi známé porty TCP a UDP komunikace. Pro simulaci této komunikace je použit protokol DNS a http.

Shrnutí kapitoly

Aplikační vrstva je horní vrstvou obou síťových modelů, její úlohou je zajistit rozhraní mezi aplikace mi spuštěnými na jednotlivých zařízeních, které potřebují komunikovat po počítačové síti a nižšími vrstvami, které zprostředkovávají síťové a přenosové funkce. Aplikační vrstva modelu TCP/IP v sobě zahrnuje funkcionalitu všech tří horních vrstev OSI modelu. Prezentační vrstva OSI modelu definuje úlohu vlastního kódování a konverze aplikačních dat. Úkolem relační vrstvy je implementovat funkce udržování dialogu mezi zdrojovou a cílovou aplikací. Podstatou je udržovat dialog, v případě chyb jej obnovit či ukončit.
Mezi nejznámější aplikační protokoly rodiny TCP/IP protokolů patří protokoly zajišťující výměnu uživatelských dat a nezbytných informací pro chod aplikací využívajících internet Mezi tyto protokoly patří: Hypertext Transfer Protocol (HTTP) pro přenos dat do webových prohlížečů, Domain Name Service Protocol (DNS) pro překlad známých Internetových adres na IP adresy, Simple Mail Transfer Protocol (SMTP) pro odesílání elektronické pošty, Telnet protokol pro vzdálený konzolový přístup na vzdálené zařízení a File Transfer Protocol (FTP) pro přenos souborů mezi koncovými zařízení.

Hlavním úkolem protokolu aplikační vrstvy je výměna dat mezi aplikacemi běžícími na zdrojovém a cílovém zařízení komunikace. Aplikační protokoly definují procesy na obou koncích komunikačního řetězce, typy vyslaných zpráv a jejich syntaxi, způsob odesílání dat a podobu očekávané odpovědi a také způsob spolupráce s nižší vrstvou síťového modelu.

Aplikační protokoly pracují na komunikačním modelu klient/server. V modelu klient/server se zařízení, které se dožaduje informace, nazývá klient a zařízení, které na požadavky odpovídá, se nazývá server. Tento model komunikace je navazován právě v aplikační vrstvě. Klient zahajuje komunikaci vysláním požadavku, na který server odpovídá.

Transportní vrstva zajišťuje segmentaci dat a jejich řízení nezbytně nutné pro různé typy komunikačních proudů. Její hlavní úlohy jsou:

- Řízení a sledování komunikace mezi aplikacemi na obou koncích.
- Segmentace dat a řízení každého segmentu
- Složení aplikačních dat zpět z proudů segmentů
- Identifikace různých typů aplikací

TCP a UDP protokoly jsou hlavními protokoly transportní vrstvy. Jejich úloha je totožná, je to řízení komunikace mezi aplikacemi. Rozdíl je však ve funkčnosti těchto protokolů. UDP je jednodušší protokol, který používá méně složitou strukturu PDU nazvanou datagramy. Datagramy jsou oproti segmentům TCP kratší, protože neobsahují sekvenční čísla a pole bytů pro potvrzení o přijetí paketu. Protokol UDP se používá v komunikaci mezi aplikacemi, které vyžadují rychlý přenos, který může být nespolehlivý. Proto není každý segment potvrzován.

Adresování pomocí portů je jedním z nejdůležitějších poslání transportní vrstvy. TCP a UDP protokoly zajišťují komunikační přenos segmentů a datagramů mezi různými

aplikacemi. Právě čísla portů jednoznačně identifikují komunikující strany. Zdrojový port je asociován s aplikací na straně klienta neboli lokálního uživatele, cílové číslo portu je asociováno s aplikací na straně hosta nebo serveru na vzdálené straně. Porty rozdělujeme do třech skupin. Tzv. dobře známé porty (well known ports) jsou čísla přidělovaná službám nebo aplikacím, které dobře známe jako např. poštovní služby či HTTP. Registrované porty jsou přiřazeny uživatelským procesům nebo aplikacím a dynamické nebo soukromé porty jsou klientům přiřazeny, když inicializují komunikaci.

Kontrolní otázka 9.1

Co je úkolem aplikační vrstvy?

Kontrolní otázka 9.2

Jakou funkcionalitu prezentační a relační vrstvy je třeba zajistit?

Kontrolní otázka 9.3

Co má na starost aplikační protokol?

Kontrolní otázka 9.4

Jakým způsobem komunikují aplikační protokoly?

Kontrolní otázka 9.5

Popište proces přidělování IP adresy DHCP serverem.

Kontrolní otázka 9.6

Co je úkolem DNS serveru?

Kontrolní otázka 9.7

Popište komunikační princip odesílání a přijímání elektronické pošty.

Kontrolní otázka 9.8

Co je charakteristické pro FTP připojení?

Kontrolní otázka 9.9

Co je úlohou transportní vrstvy?

Kontrolní otázka 9.10

Jaký je rozdíl v použití protokolů TCP a UDP?

Kontrolní otázka 9.11

Co jsou to dobře známé porty?

Kontrolní otázka 9.12

Popište princip komunikace prostřednictvím portů.

10. IP ADRESOVÁNÍ

Čas ke studiu: 2 hodiny

Cíl Po prostudování této kapitoly budete umět

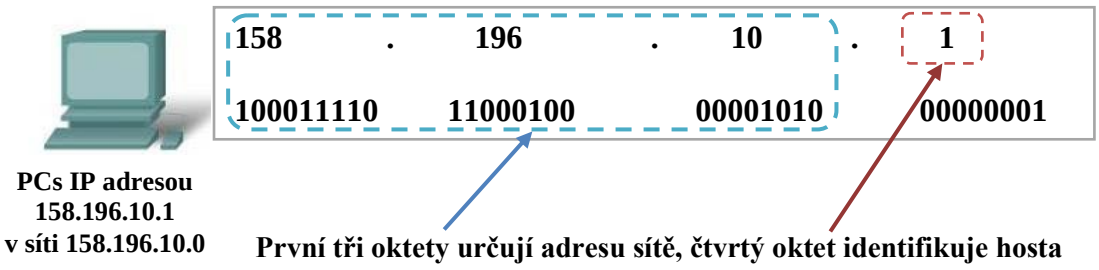
- Charakterizovat formát IP adresy
- popsat mechanismus výpočtu adresy sítě
- rozdělit IP adresy do různých skupin
- popsat proces broadcast, multicast a unicast komunikace
- definovat význam síťové masky
- popsat princip adresace podsítí

Výklad

V našem výkladu o komunikaci v počítačových sítích se dostáváme hlouběji do jednotlivých vrstev. Než si podrobně popíšeme komunikaci na úrovni síťové vrstvy, je třeba vysvětlit některé nezbytné pojmy týkající se adresace v sítích. Tato kapitola se tedy podrobně věnuje charakteristice a rozdělení IP adres a dalšími nezbytnými pojmy adresování.

10.1 Definice IP adresy

Každé zařízení v síti musí být jednoznačně rozeznatelné. Pro doručení jakýchkoliv dat danému zařízení potřebuje znát jeho adresu. V počítačové síti se používá termín IP adresa a má jednoznačný 32 bitový formát (IP protokol verze 4). Protože počítače rozumí binární formě dat reprezentovanou nulami a jedničkami, musíme pro člověka a jeho snazší orientaci formát IP adresy převést na dekadický. Proto např. IP adresu *10001111011000100000010100000001* raději uvádíme ve formátu 158.196.10.1. Jak vidíte tuto adresu, která má v binární podobě délku 32 bitů, před samotnou konverzí na dekadický formát rozdělíme na čtyři oktety. Každý oktet je oddělen tečkou a jeho hodnota se logicky může pohybovat v rozmezí 0 až 255.



Obr. 10.1 binární a dekadický tvar IP adresy

Z obrázku je patrno, že pokud adresa sítě bude v tomto tvaru, máme poslední oktet vyhrazený na adresu hostů. Máme tedy k dispozici 255 adres, přičemž poslední adresu ještě musíme rezervovat pro tzv. *broadcast* adresu (viz další výklad kapitoly). Co když ale budeme potřebovat zapojit do sítě více než 254 počítačů? To samozřejmě lze, jen adresa sítě bude jiná a adresy jednotlivých hostů budou zasahovat již do třetího oktetu. S tímto mechanismem úzce souvisí maska sítě a tu si popíšeme v následujících odstavcích.

Korespondenční úkol 10.1 – zjištění IP adresy

Příkazem *ipconfig* v příkazovém řádku zjistěte IP adresu počítače, na kterém právě pracujete. Pro získání více údajů o připojení vašeho PC zadejte *ipconfig /all*

10.2 Převody mezi binární a desítkovou soustavou

Pro vnímání adresy v dekadické podobě je nutné znát mechanismus převodu mezi oběma soustavami a to oběma směry. Nejprve si popíšeme převod z binární soustavy na dekadickou. Vystačíme si s osmi bity, neboť jsme si 32 bitovou IP adresu rozdělili na čtyři stejné osmibitové oktety. Mechanismus převodu je naznačen na obrázku 10.2, přičemž jednotlivým bitům je udávána váha bitu jako mocnina $2^{\text{POZICE BITU}}$, kde pozice bitu je číslována od nuly směrem zprava doleva. Pro výsledný dekadický tvar se sečtou váhy bitu na jednotlivých jedničkových pozicích.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	exponent
128	64	32	16	8	4	2	1	hodnota váhy bitu
1	1	0	0	0	1	0	0	binární vyjádření oktetu
128 +	64 +	0 +	0 +	0	+ 4	+ 0	+ 0	součet vah jedničkových bitů
196								decimální vyjádření oktetu

Obr. 10.2 převod binární adresy na dekadickou

Převod z dekadické soustavy na binární využívá obdobného mechanismu. Rovněž zde hrají úlohu váhy bitu, které odečítáme od dekadické hodnoty čísla. V případě že dekadické číslo je větší nebo rovno váze bitu, váhu bitu od čísla odečteme a nastavíme na příslušné pozici jedničkový bit.

V případě že dekadické číslo je menší než váha bitu, tak na příslušnou pozici nastavíme nulový bit a pokračujeme v porovnání dekadické hodnoty s nižší váhou. Takto pokračujeme, dokud dekadickou hodnotu úplně nevynulujeme. V případě že je již dekadická hodnota nulová, napíšeme na zbývající pozice osmibitového čísla nuly.

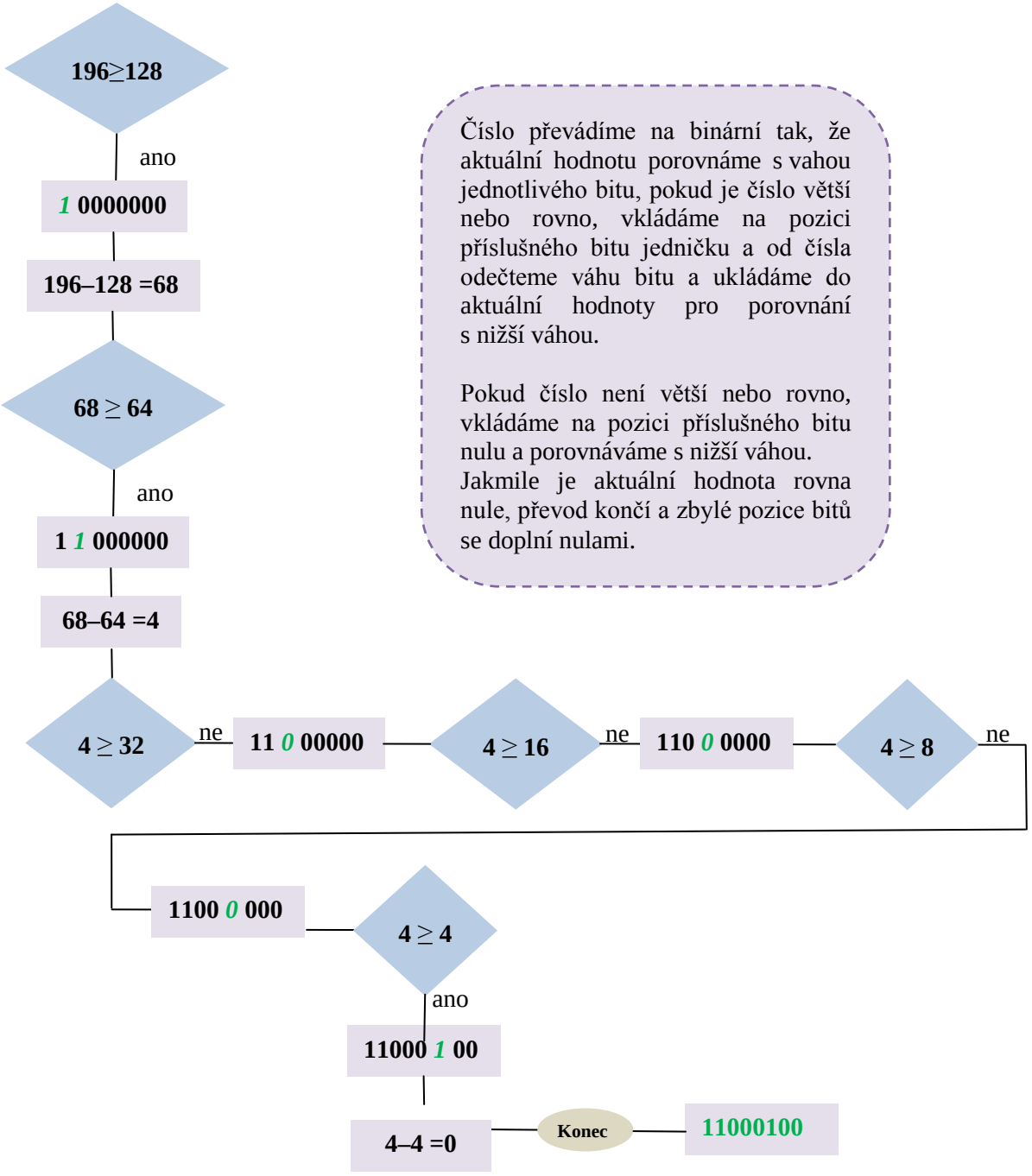
Korespondenční úkol 10.2 – převody mezi soustavami

Uvedený postup pro osmibitové číslo je obecný. Tímto způsobem převádíme jakkoliv velké číslo, pouze hodnoty vah budou samozřejmě vyšší a opět budou rovny mocnině dvojky podle pozice bitu.

Převeďte na dekadickou podobu desetibitové číslo 1010101101

Pojem k zapamatování: Používané číselné soustavy

Kromě dekadické a binární soustavy budeme v počítačových sítích využívat také soustavu hexadecimální neboli šestnáctkovou. Tato soustava využívá číslic 0 – 9 a písmen A- F pro vyjádření hodnot deset až patnáct. Tuto soustavu využívají MAC adresy.



Obr. 10.3 převod dekadického čísla na binární

10.3 Původní rozdělení IP adres

V raných dobách počítačových sítí byly IP adresy rozděleny do tzv. tříd. Jednotlivé třídy IP adres se lišily velikostí adresovatelného prostoru neboli počtem možných IP pro přidělení. Původní myšlenka tedy byla přiřazovat obrovským organizacím právě adresní prostor pro mnoho počítačů a malým organizacím menší adresní prostor. Jenže později se ukázalo, že při velkém nárůstu počítačových sítí by se počet možných přidělených IP adres rychle vyčerpal a hlavním problém byl také fakt, že opravdu vysoký počet IP adres by byl nevyužitý přesto, že by se již nikomu nemohl přiřadit. Důvod proč tomu tak je si vysvětlíme po následujícím rozdělení IP adres do tříd. IP adresy jsou rozděleny do pěti tříd A-E (viz. Tab.10.1)

Tab. 10.1 Třídy IP adres

Třída adres	Dekadický tvar prvního oktetu	Bitový rozsah prvního oktetu	Části IP adresy (N- síť, H –host)	Defaultní maska	Počet možných sítí a jejich hostů
A	**1-127**	**00000000 - 01111111**	**N.H.H.H**	**255.0.0.0**	**128 sítí 16777214 hostů/síť**
B	**128-191**	**100000000 - 10111111**	**N.N.H.H**	**255.255.0.0**	**16384 sítí 65534 hostů/síť**
C	**192-223**	**11000000 - 11011111**	**N.N.N.H**	**255.255.255.0**	**2097150 sítí 254 hostů/síť**
D	**224-239**	**11100000 - 11101111**	Multicast adresy		
E	**240-255**	**11110000 - 11111111**	Experimentální adresy		

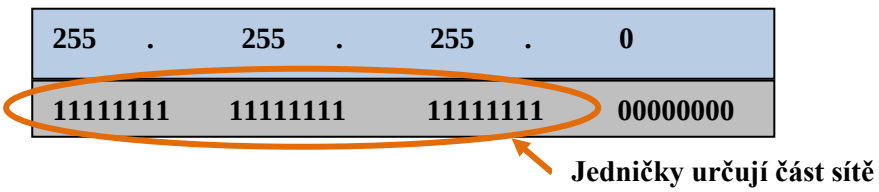
Původní myšlenka tedy byla vyhradit 128 rozsáhlých sítí pro hodnotu prvního oktetu 0-127. Zbývající tři oktety neboli 24 bitů je vyhrazeno pro IP adresy sítě. Další třídy adres umožňovalo vytvořit další velikosti sítě pro 65534 a 254 hostů. Z tabulky vidíme, že počet takových sítí by mohl být 2097150.
Jak už ale bylo řečeno, počet sítí by po určité době nestačil, zvláště ze zbytečně vyplýtvaného prostoru třídy A. Rovněž počet nikdy nevyužitých IP adres je značný. Pro příklad velká organizace ze třídy A bude disponovat milionem připojených zařízení, tak zbývajících 15777214 adres bude nevyužitých a tyto adresy nebude možno již nikomu přidělit, protože síťovou adresu ve tvaru N.H.H.H bude mít pod svou správou již tato společnost. Proto se v devadesátých letech dvacátého století upustilo od toho typu adresování a na řadu přišlo takzvané **beztřídní adresování** *(classless addressing)*, které umožní využít celý adresní prostor pomocí podsítí určenou tzv. maskou sítě.

Pojem k zapamatování: maska sítě

Maska sítě je opět 32bitové číslo ve tvaru IP adresy rozdělené do 4 oktetů. Maska sítě má široký rozsah uplatnění, pomocí ní rozdělujeme sítě na podsítě, určujeme příslušnost ke konkrétní síti, počítáme adresy sítě atd. S maskou sítě pracují síťová zařízení, pomocí masky sítě se provádí směrování a mnoho dalších událostí.

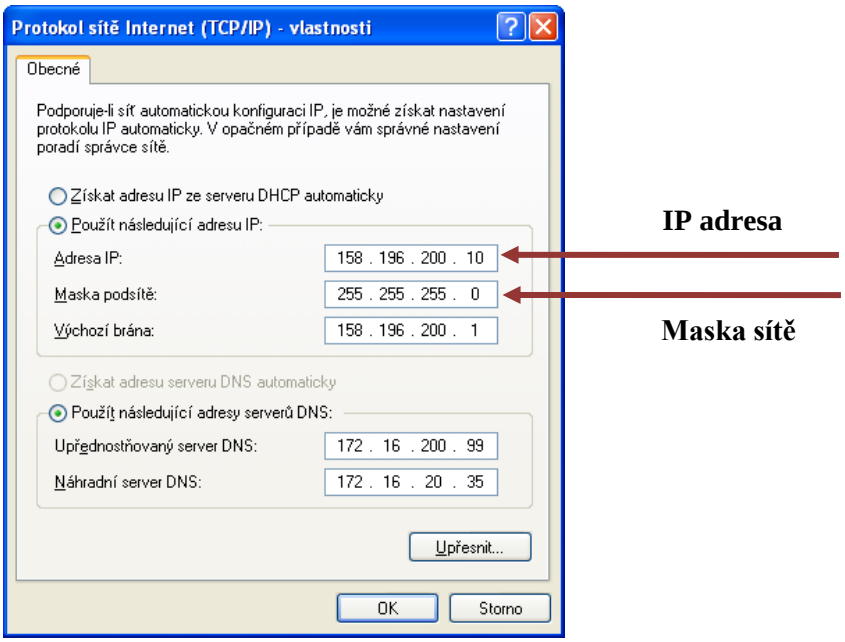
10.4 Maska sítě

Mnoho uživatelů vyplňuje při nastavení IP adresy masku sítě (pokud automaticky nepřiděluje DHCP), aniž by vůbec věděli, co konkrétní údaj znamená. Nejčastěji zadáváme masku ve tvaru *255.255.255.0* viz. Obr. 10.5.
Abychom nutné nastavení IP adresy hosta a jeho příslušnost ke konkrétní síti pochopili úplně, je třeba vysvětlit přesný tvar a použití masky sítě. Jak je vysvětleno výše, určitá část IP adresy definuje síť (např. první tři oktety) a určitá část definuje hosta (např. poslední oktet).



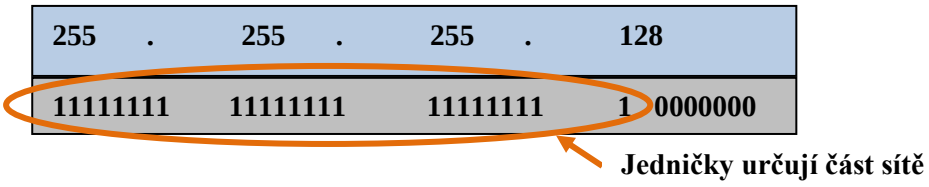
Obr.10.4 Maska sítě

A protože chceme vědět, do které sítě host patří, potřebuje znát masku sítě. Na obrázku 10.4 je uvedena maska sítě se samými jedničkami v prvních třech oktetech a nulovým posledním oktetem. V principu to znamená, že první tři oktety určují IP adresu sítě (jedničky) a čvrtý oktet určuje adresu hosta (nuly). Takže tam kde vidíme v masce nuly, je prostor pro hosta. Z této masky vyplývá, že poslední oktet je věnovaný adresám hostů, to je dohromady 256 adres, dvě adresy jsou však vyhrazeny, jedna pro adresu sítě a druhá (samé jedničky) pro adresu tzv. *broadcast*. Pro IP adresu z obrázku 10.5 by to znamenalo, že *158.196.200.0* je adresa sítě, *158.196.200.255* adresa broadcast a zbývající adresy 1 -254 jsou určeny pro hosty. Mezi těmito 254 adresami máme rovněž adresu pro výchozí bránu tzv. *Default gateway*.



Obr. 10.5 Manuální nastavení IP adresy a masky

Ne vždy ale bývá maska tak jednoduchá. Například máme-li masku *255.255.255.128*, musíme opět určit pomocí nul a jedniček, která část je síťová a kolik prostoru zbývá pro hosty. Postupujeme tedy následujícím způsobem:



Obr. 10.6 Určení adresy sítě a adresní části pro hosta

Kdybychom tedy změnili takto masku z předchozího příkladu na obrázku 10.5, tak zjistíme, že počet nul v masce sítě určuje adresní prostor pro hosty a ten je nyní zkrácen na 127. A také znova *158.196.200.0* bude adresa sítě a nyní již *158.196.200.127* adresa pro *broadcast* a zbývajících 125 adres je určeno pro hosty. Zároveň zjistíme, že následující *158.196.200.128* může být adresa další sítě, která bude mít k dispozici stejný počet hostů. Touto maskou jsme tedy rozdělili adresní prostor *158.196.200.0 – 158.196.200.255* na dvě sítě a to je podstata masky a tvorby podsítí.

Pojem k zapamatování: Prefix síťové masky

Můžeme se setkat i se zkráceným zápisem síťové masky, tzv. prefixem. Ten se určuje počtem jedničkových bitů v masce, takže prefix /24 odpovídá masce 255.255.255.0, prefix /25 odpovídá masce 255.255.255.128, /26 odpovídá masce 255.255.255.192 atd.

Korespondenční úkol 10.3 – prefix masky

Procvičte si na několika příkladech prefix masky. Jaké masce odpovídá prefix /30 nebo např. prefix /21 ?

V souvislosti s prefixem masky je vhodné také znát tzv. bitové šablony. Obdobně jako váhy bitu při převodu mezi soustavami bychom měli znát jednotlivé bitové šablony masky sítě.

10000000 = 128	**11000000=192**	**11100000= 224**	**11110000=240**
11111000= 248	**11111100=252**	**11111110=254**	**11111111=255**

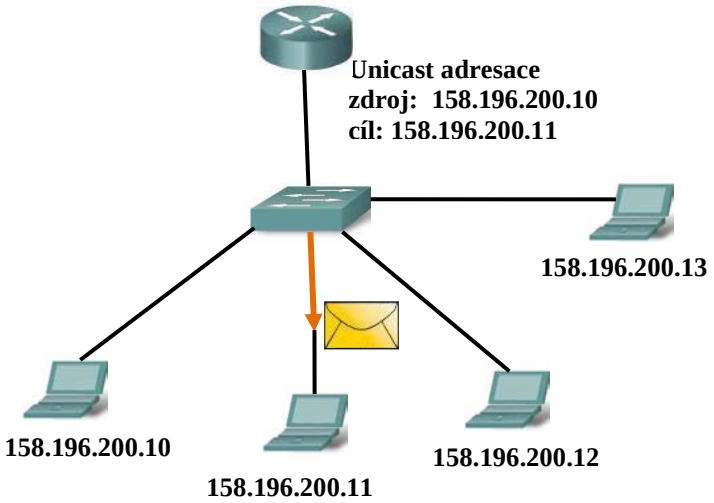
Obr. 10.7 bitové šablony masky sítě

10.5 Unicast, Broadcast a Multicast adresace

Při adresování v počítačových sítích se používají tři typy komunikace – unicast, broadcast a multicast:

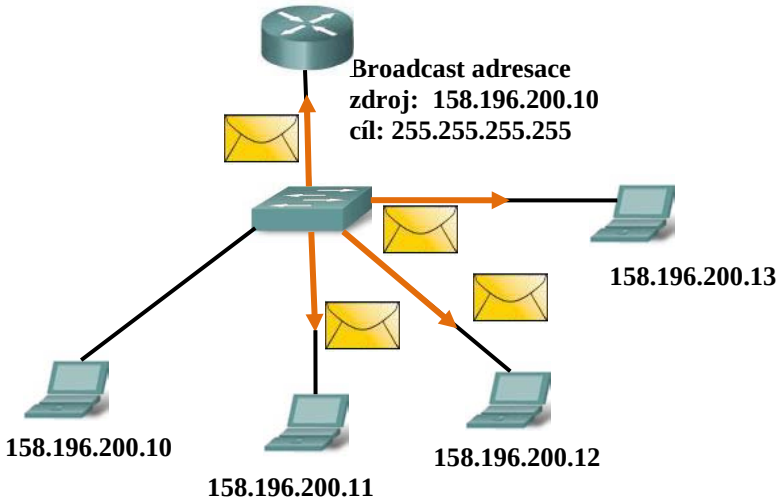
- Unicast – proces vyslání paketu jednomu hostu
- Broadcast – proces vyslání paketu všem hostům v síti
- Multicast – proces vyslání paketu vybrané skupině hostů

Unicast komunikace je pro uživatele nejpochopitelnější. Jeden host posílá paket svému vybranému cíli. Zdroj I cíl je definovaný jednoznačně svou IP adresou.



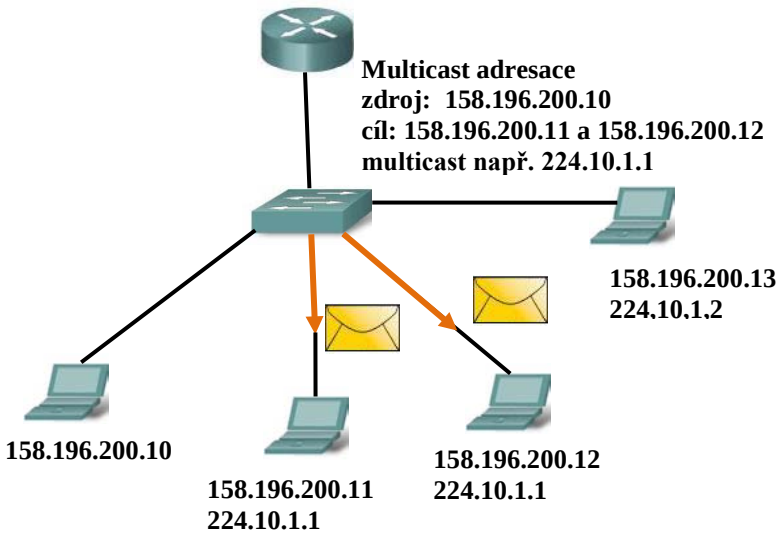
Obr. 10.8 komunikace v síti typu unicast

Dalším typem komunikace je typ broadcast. V předchozí části jsme si popisovali formát IP adresy a v každém adresovém rozsahu jsme museli počítat se dvěma vyhrazenými adresami. První byla pro adresu sítě a druhá pro broadcast komunikaci. Broadcast adresa má vždy ve svém rozsahu same jedničky.
Broadcast komunikace v principu pracuje tak, že daný paket je vyslán všem účastníkům sítě. Broadcast komunikaci nejčastěji využívají protokoly, které něco hledají, neznají adresu svého cíle, ale mají pro něj zprávu. Obsah paketu je tvořen informacemi, které cílový adresát pozná a ostatní účastníci paket zahazují. S broadcast paketem jsme se setkali například při DHCP komunikaci v předchozí kapitole. Přestože účastníci paket, jež pro ně není určen, zahazují, broadcastových paketů je opravdu mnoho a zahlcují provoz na síti. Proto bylo třeba zavést ochranné mechanismy pro šetření zátěže sítě jako např. životnost paketů.



Obr. 10.9 komunikace v síti typu broadcast

Třetím typem komunikace je multicast. Paket je ze zdrojové adresy posílán pomocí tzv. Multicastové adresy z adresového rozsahu 224.0.0.0 – 239.0.0.0 viz. tabulka tříd IP adres. Tyto IP adresy jsou mapovány na MAC adresy cílové skupiny uživatelů a je úkolem druhé vrstvy ISO/OSI modelu zajistit doručení paketu. Multicast komunikace je často využívaná v situacích kdy jeden zdroj vysílá data vice příjemcům najednou (streamovaná videoconference, chat, atd.).



Obr. 10.10 komunikace v síti typu multicast

10.6 Speciální typy IP adres

Některé IP adresy jsou vyhrazeny pro speciální účely. V minulých kapitolách jsme si představili adresní rozsah pro multicastovou komunikaci. Za tímto rozsahem vidíme v tabulce 10.1 zbylý adresní prostor, který je připraven pro budoucí účely. Dnes je tento adresní prostor 240.0.0.0-255.255.255.254 označován jako experimentální blok adres pro testovací účely a výzkum.
Adresní rozsah určený pro běžné použití v počítačových sítích rozdělujeme ještě na veřejný a privátní. Veřejné adresy jsou používané pro zařízení a server, které jsou veřejně přístupné z internetu, oproti tomu privátní adresy jsou využívány tam, kde jsou uživatelé před veřejnou internetovou službou skryti, nebo k ní mají omezený přístup. V dnešní době se v mnoha případech využívá tzv. překlad adres z privátní na veřejnou.

Pojem k zapamatování: NAT – Network Address Translation – překlad adres

Překlad adres umožňuje celou privátní síť s mnoha počítači s privátní adresou zaštítit jednou veřejnou IP adresou nebo rozsahem veřejných adres. Pokud počítač z privátní sítě komunikuje ven ze své sítě, je mu překladem adres poskytnuta veřejná adresa.

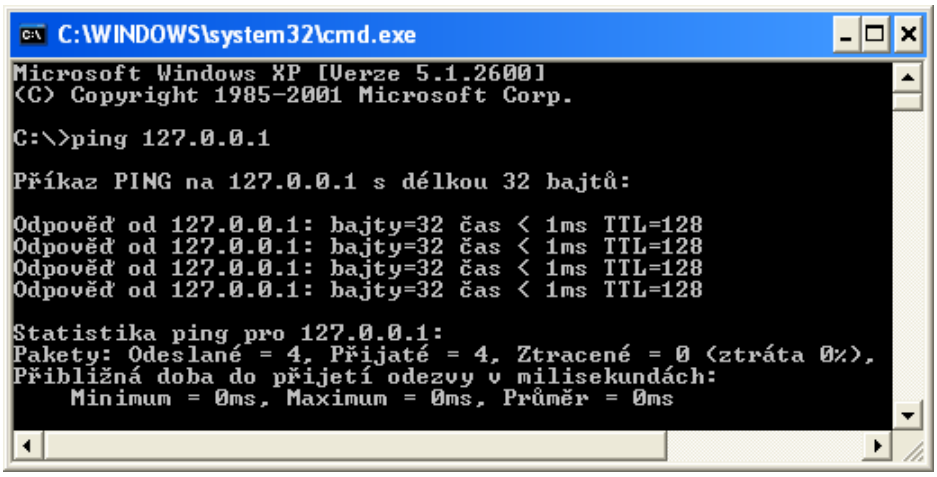
Privátní adresy se vyskytují v těchto blocích (závorce vidíme zkrácený zápis rozsahu pomocí adresy sítě a její masky):

- 10.0.0.0 - 10.255.255.255 (10.0.0.0 /8)
- 172.16.0.0 - 172.31.255.255 (172.16.0.0 /12)
- 192.168.0.0 - 192.168.255.255 (192.168.0.0 /16)

Mezi speciální typy adres patří nám již velmi dobře známá adresa broadcastu 255.255.255.255. Přesně opačná adresa 0.0.0.0 má také svůj význam a to při směrování. Používá se na routerech k definici defaulní cesty. Poslední speciální adresou, o které se zmíníme, je adresa 127.0.0.1. Touto adresou se označuje tzv. *loopback* neboli cesta zpět na sebe sama. Neposíláme tedy pomocí této IP adresy žádné pakety do sítě. Adresu používáme, pokud chceme zjistit zdali naše vlastní konfigurace je v pořádku.

Korespondenční úkol 10.3 – ping 127.0.0.1

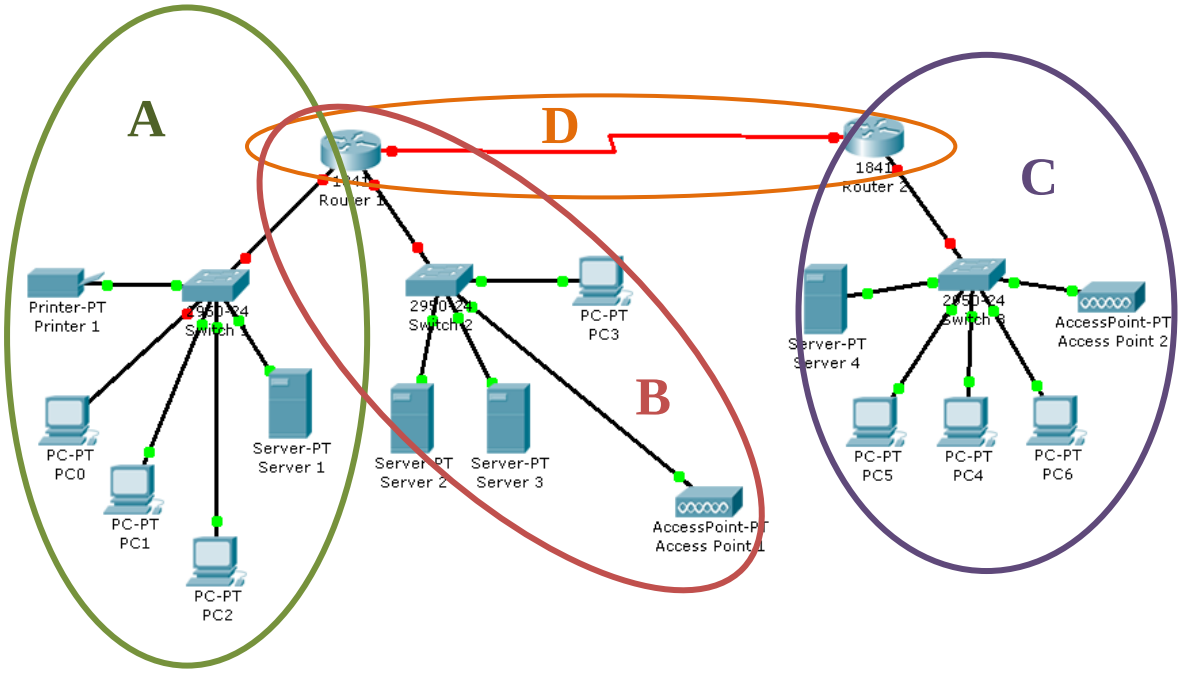
Otevřete příkazový řádek neboli *Command prompt* a napište do něj příkaz *ping 127.0.0.1*



Zkontrolujte tak, zda-li je protokol stack TCP/IP na vašem PC správně nakonfigurován.

10.7 Adresování podsítí

Nyní si praktickém příkladu ukážeme, jak můžeme dokonale pomocí síťové masky využít adresní prostor. Na obrázku 10.11 vidíme příklad sítě střední organizace, která má dvě sítě připojené k jednomu routeru a třetí síť připojenou k routeru druhému. Nesmíme zapomenout na fakt, že routery jsou připojené mezi sebou seriovým rozhraním a tvoří tak ve skutečnosti malou síť s dvěma IP adresami, nicméně musíme s touto sítí při návrhu adresního prostoru počítat. Ve skutečnosti tedy administrujeme 4 počítačové sítě znázorněné na obrázku písmeny A-D. Z potřeb organizace vyplynulo, že síť A je největší, obsahuje počítače, servery a síťové tiskárny a dohromady i s nějakou rezervou do budoucna potřebujeme adresovat 100 zařízení. Síť B je o polovinu menší, ale zaměstnanci si nosí mobilní zařízení, které přes Wifi access point připojují do sítě, a tak je potřeba vyhradit celkem 60 adres. Síť C obsahuje útvar ve vzdálené budově, kde je zapotřebí počítat se 30 adresami. Nesmíme zapomenout již zmíněnou síť D mezi dvěma routery se dvěma IP adresami.



Obr. 10.11 Topologie sítě pro příklad adresování podsítí

Berme příklad jako ilustrativní, s tím že máme k dispozici libovolný adresní prostor od adresy 158.196.200.0. Bez použití podsítí bychom si nemuseli lámat hlavu s maskou sítě a vytvořili bychom 4 stejně velké sítě 158.196.200.0 – 158.196.203.0. Každá síť by tak měla k dispozici 254 adres pro připojená zařízení, adresu sítě a broadcastu. Jenže v požadavcích na adresní prostor vidíme daleko menší počet zařízení a co teprve u sítě D s dvěma IP adresami, zde zůstane navždy nevyužitých 252 adres.

Tab. 10.2 Adresování bez použití podsítí

Síť	Adresa	maska	Adresní prostor	Počet dostupných adres	Počet požadovaných IP adres	Nevyužito
A	158.196.200.0	/24	158.196.200.0 - 158.196.200.255	254	100	154
B	158.196.201.0	/24	158.196.201.0 - 158.196.201.255	254	60	194
C	158.196.202.0	/24	158.196.202.0 - 158.196.202.255	254	30	224
D	158.196.203.0	/24	158.196.203.0 - 158.196.203.255	254	2	252
celkem						**824**

Z tabulky je patrno, že při návrhu adresního prostoru bez použití podsítí bychom ztratili až 824 IP adres a to je při požadovaných 202 IP adres přímo hrozivé číslo. Z příkladu tedy vyplývá, že bylo nutné najít efektivní způsob jak adresní prostor využít.

Tento problém řeší použití masky sítě a my si nyní ukážeme, jakým způsobem ji použijeme a kolik IP adres využijeme na naše 4 sítě a samozřejmě z uvedeného postupu vyplyne kolik IP adres ušetříme. Zkušený síťový administrátor na tuto otázku odpoví již teď, protože všechny 4 sítě A-D umístíme do jednoho adresního prostoru 158.196.200.0 – 158.196.200.255, ušetříme tedy minimálně celý adresní rozsah 158.196.201.0 – 158.196.203.255 !

Vše provedeme následujícím způsobem:

Budeme postupovat od největší sítě k nejmenší, což v našem příkladě znamená, že pořadí sítí zůstane zachováno. Síť A tedy potřebuje adresní prostor pro 100 zařízení. Z vlastnosti masky sítě víme, že maska *255.255.255.128* neboli */25*nám adresní prostor rozdělí na dvě poloviny se 128 možnými adresami. My potřebujeme adres 100, takže nám tato maska bezesporu vyhovuje.

Síť A tedy bude mít opět adresu 158.196.200.0, ale pozor s maskou /25.

To znamená, že adresní prostor již bude pouze do adresy 158.196.200.127 (broadcast tohoto prostoru) a na adrese 158.196.200.128 již bude síť B. Otázkou zůstává, jakou masku sítě pro ni zvolíme. Masku /25 bychom použít samozřejmě mohli, ale tím bychom si vyplýtvali prostor pro další sítě. Když využijeme v masce sítě další bit, zvětšíme masku sítě na 255.255.255.192 (viz. Bitové šablony na obrázku 10.7), dosáhneme tak zmenšení adresového prostoru na 64 IP adres.

Síť B tedy bude mít adresu 158.196.200.128 ale s maskou /26 a další síť začne na adrese 158.196.200.192. Už se to pomalu začíná rýsovat, že? Maska sítě tedy vždy rozděluje adresní prostor na více sítí, záleží na nás, jak ji použijeme, resp. na požadavcích sítě.

Vraťme se k našemu příkladu, máme adresní prostor pro síť A, B a zbývá nám ještě 64 IP adres, tedy adresní prostor 158.196.200.192 s maskou /26. Pokud by síť C požadovala více než 32 IP adres museli bychom zbylý adresní prostor použít celý, avšak síť C potřebuje 30 IP adres, můžeme tedy síť ještě dělit. Maska tedy bude /27 neboli 255.255.255.224 a rozdělí nám zbylý prostor na dvakrát 32 adres.

Síť C tedy bude v rozsahu 158.196.200.192 – 158.196.200.223 s maskou /27. Zbyl nám dostatečný prostor na síť D, kde potřebujeme jenom 2 IP adresy (pozor ve skutečnosti 4 – ještě adresa sítě a broadcastu). Je na nás, jestli síti D necháme masku /27, tedy 32 IP adres, 4 využijeme a 28 zahodíme, a nebo budeme síť dále dělit. V podstatě to není nutné, protože nám již mnoho IP adres nezbývá, ale nutno podotknout, že adresní prostor byl rozdělen dle počátečních požadavků.

Pojem k zapamatování: Dělení adresního prostoru

Nejdůležitějším bodem návrhu je definování požadavku počtu IP adres. Při adresování maskou podsítě nedochází k plýtvání IP adresami, a proto musíme při návrhu sítě počítat s požadavky do budoucna.

Tab. 10.3 Adresování s použitím podsítí

Síť	Adresa	maska	Adresní prostor	Počet dostupných adres	Počet požadovaných IP adres	Nevyužito
A	158.196.200.0	/25	158.196.200.0 - 158.196.200.127	126	100	26
B	158.196.200.128	/26	158.196.200.128 - 158.196.200.191	62	60	2
C	158.196.200.192	/27	158.196.200.192 - 158.196.200.223	30	30	0
D	158.196.200.224	/27	158.196.200.224 - 158.196.200.255	30	2	30
celkem					**58**	

Z příkladu je jednoznačně prokazatelná úspora adresního prostoru při použití síťové masky. Nutno podotknout, že u sítě C jsme využili adresní prostor beze zbytku, nelze tudíž v budoucnu žádat pro

tuto síť další IP adresy, což například u sítě A ještě lze. To vše se však musí zvážit při samém počátku návrhu.

Ke kapitole 10 je přiřazena demonstrační animace č. 8 a č. 9

Animace č. 8 obsahuje nastavení IP adres rozhraní routeru, ukázku konfigurace v příkazovém řádku a umožnění přístupu na routeru z PC pomocí příkazu telnet. Animace č. 9 obsahuje ukázku nastavení IP adres pro všechny navržené sítě, ukázku konfigurace IP adres a masek podsítí na routeru a zobrazení všech dostupných sítí.

Shrnutí kapitoly

IP adresa a má jednoznačný 32 bitový formát (IP protokol verze 4). Adresu s tímto formátem v binární podobě před samotnou konverzí na dekadický formát rozdělíme na čtyři oktety. Každý oktet je oddělen tečkou a jeho hodnota se logicky může pohybovat v rozmezí 0 až 255. Pro vnímání adresy v dekadické podobě je nutné znát mechanismus převodu mezi oběma soustavami a to oběma směry.

IP adresy jsou rozděleny do pěti tříd A-E, tyto třídy se liší počtem možných sítí a jejich hostů. Tomuto rozdělení se říká třídní rozdělení. Třídní adresování přestalo vyhovovat z důvodu velkého nárůstu počítačových sítí a nevyužitelnosti velkých bloků adres. V devadesátých letech dvacátého století upustilo od toho typu adresování a na řadu přišlo takzvané **beztřídní adresování** *(classless addressing)*, které umožní využít celý adresní prostor pomocí podsítí určenou tzv. maskou sítě. Maska sítě je opět 32bitové číslo ve tvaru IP adresy rozdělené do 4 oktetů. Maska sítě má široký rozsah uplatnění, pomocí ní rozdělujeme sítě na podsítě, určujeme příslušnost ke konkrétní síti, počítáme adresy sítě atd. S maskou sítě pracují síťová zařízení, pomocí masky sítě se provádí směrování.
Při adresování v počítačových sítích se používají tři typy komunikace – unicast, broadcast a multicast: Unicast – proces vyslání paketu jednomu hostu

- Broadcast – proces vyslání paketu všem hostům v síti
- Multicast – proces vyslání paketu vybrané skupině hostů

Některé IP adresy jsou vyhrazeny pro speciální účely. Multicastové adresy jsou z adresového rozsahu 224.0.0.0 – 239.0.0.0, adresní prostor 240.0.0.0- 255.255.255.254 je označován jako experimentální. Adresní rozsah určený pro běžné použití v počítačových sítích rozdělujeme ještě na veřejný a privátní. Veřejné adresy jsou používané pro zařízení a server, které jsou veřejně přístupné z internetu, oproti tomu privátní adresy jsou využívány tam, kde jsou uživatelé před veřejnou internetovou službou skryti, nebo k ní mají omezený přístup. V dnešní době se v mnoha případech využívá tzv. překlad adres z privátní na veřejnou. Mezi speciální typy adres patří adresa broadcastu 255.255.255.255, opačná adresa 0.0.0.0 má vůj význam při směrování. Používá se na routerech k definici defaulní cesty. Další speciální adresou je adresa loopback 127.0.0.1. Zkontrolujeme tak, zda-li je protokol stack TCP/IP na našem PC správně nakonfigurován.

Úkol k řešení 10.1 – Návrh adresního prostoru sítě

Pro síť z obrázku 10.11 realizujte stejným způsobem, jako jsme si představili ve výkladu návrh adresního prostoru za předpokladu, že sítě obsahují následující počet zařízení: A:60, B:55, C:31, D:2.

Úkol k řešení 10.2 – Návrh adresního prostoru sítě

Realizujte rozdělení adresního prostoru 192.168.5.0 /24 pro následují sítě a počty zařízení:
A: 110, B: 20, C:20, D:12, E:2, F:2

Kontrolní otázka 10.1

Jaký formát má IP adresa?

Kontrolní otázka 10.2

Popište rozdělení adresního prostoru na třídy.

Kontrolní otázka 10.3

Jaký formát má maska sítě?

Kontrolní otázka 10.4

Jaký význam má maska sítě?

Kontrolní otázka 10.5

Jaké speciální IP adresy známe?

Kontrolní otázka 10.6

Co je to NAT a k čemu se používá?

Kontrolní otázka 10.7

Jaké typy komunikace znáte?

Kontrolní otázka 10.8

Jakým způsobem a k čemu se používá komunikace typu broadcast?

11. KOMUNIKACE NA ÚROVNI SÍŤOVÉ VRSTVY

Čas ke studiu: 2 hodiny

Cíl Po prostudování této kapitoly budete umět

- popsat jednotlivé činnosti na síťové vrstvě
- popsat jednotlivé části paketu
- charakterizovat IP protokol a jeho činnost
- nastavit defaultní bránu sítě a vysvětlit její význam
- definovat princip směrování
- vysvětlit význam směrovací tabulky

Výklad

V předchozí kapitole jsme detailně probrali IP adresy, takže máme přehled o nejpodstatnějších pojmech, které jsou základem pro adresování a směrování. Nyní máme připravenou půdu pro vysvětlení principu směrování paketu, což je úloha síťové vrstvy modelu. Seznámíme se mimo jiné se dvěma důležitými pojmy, což je cesta v síti a směrovací tabulka.

11.1 Úlohy síťové vrstvy

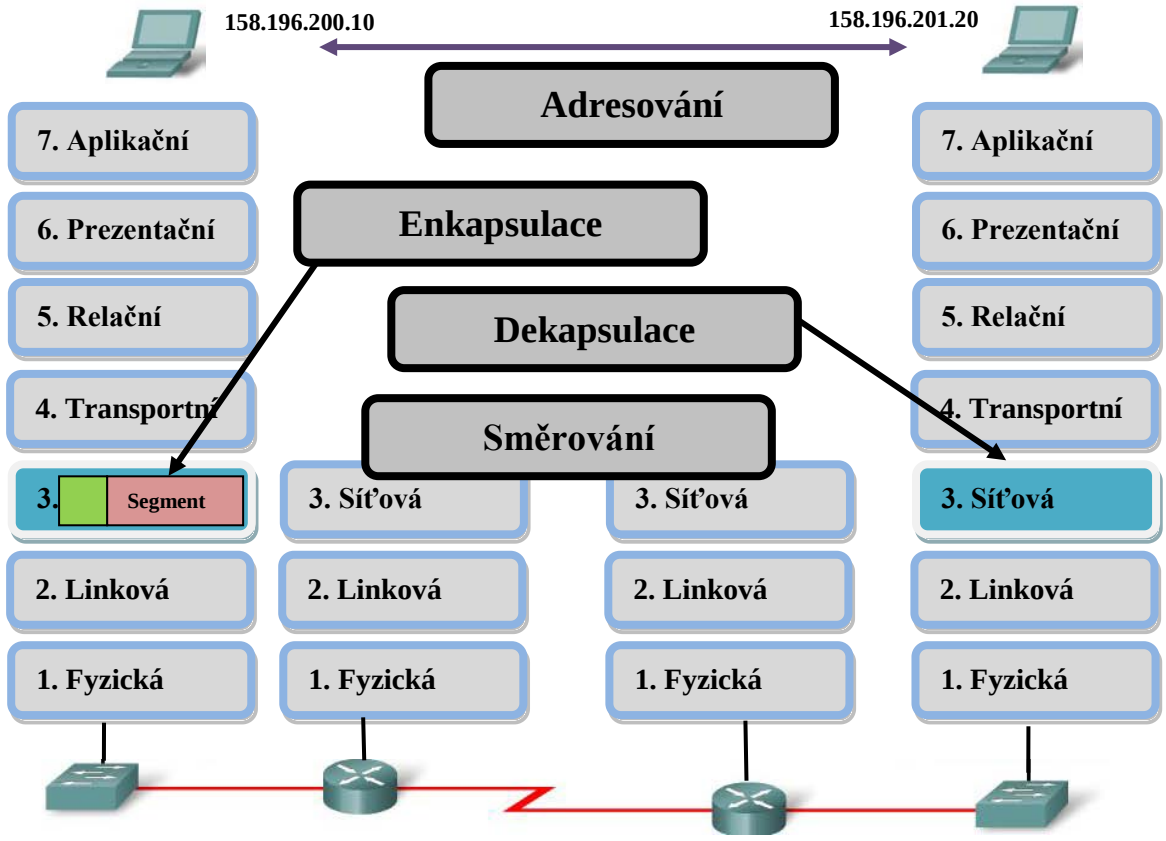
Síťová vrstva je třetí vrstvou OSI modelu a jejím hlavním úkolem je zajistit výměnu dat mezi jednotlivými uživateli sítě. Zajištění správného chodu této komunikace se skládá ze čtyř základních procesů:

- Adresování
- Enkapsulace
- Směrování
- Dekapsulace

Adresování
Mechanismus adresování koncových zařízení je zajištěn IP adresou, každé zařízení v síti je identifikováno jednoznačnou IP adresou, jejíž IPv4 (IP protokol verze 4) jsme si definovali v minulé kapitole.

Enkapsulace
Síťová vrstva zajišťuje proces zabalení segmentu z nadřazené transportní vrstvy. K tomuto segmentu je přidána IP hlavička, proces enkapsulace tak vytvoří paket síťové vrstvy. IP hlavička obsahuje zdrojovou a cílovou IP adresu komunikačního procesu. Po ukončení enkapsulačního procesu je paket předán nižší vrstvě síťového modelu.

Směrování
Zdrojové a cílové zařízení nejsou obvykle ve stejné síti, proto síťová vrstva musí zajistit směrování paketů do správného segmentu sítě. To se provádí pomocí zprostředkujících zařízení, kterým říkáme routery. Proto se pro termín směrování často používá jeho anglický ekvivalent *routing*. Router je zařízení na třetí vrstvě síťového modelu a proto je vždy na síťové cestě prováděn proces enkapsulace a dekapsulace. Pouze segment, neboli informační jednotka transportní jednotky zůstane nedotčena, zatímco paket je měněn podle toho, jak je směrován síťovou vrstvou.



Obr. 11.1 Úlohy síťové vrstvy

Dekapsulace
Routery a v samotném závěru také cílové zařízení provádějí na síťové vrstvě proces dekapsulace, neboli samotné rozbalení paketu. Zatímco router musí rámec znovu zabalit a vyslat dále do sítě, cílové zařízení rozbalený rámec zpracuje a jeho segment vyšle nadřazené transportní vrstvě.

11.2 Charakteristika IP protokolu

Protokol síťové vrstvy obecného OSI modelu může být různý. Zde uvádíme krátký seznam možných protokolů:

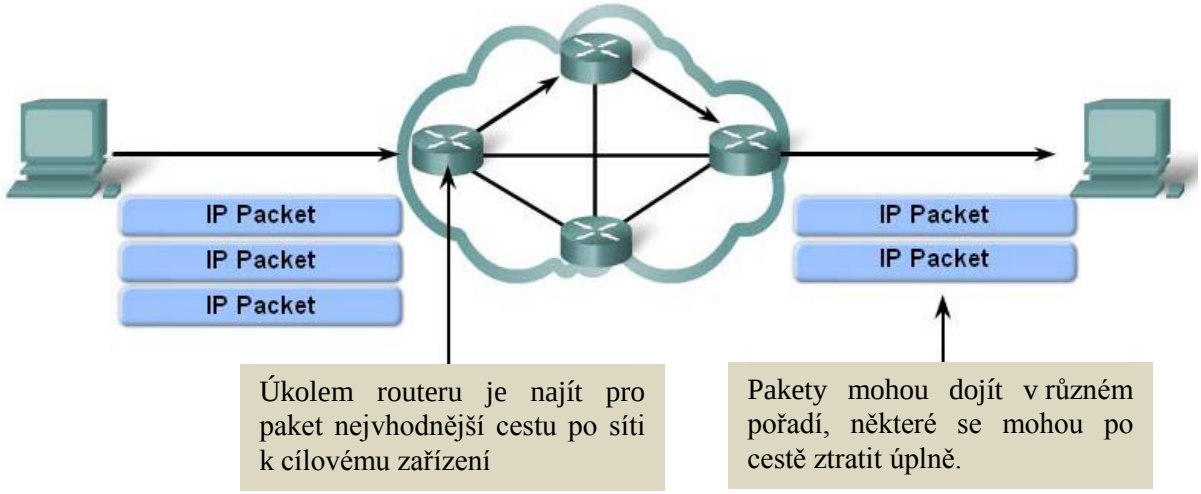
- Internet Protocol version 4 (IPv4)
- Internet Protocol version 6 (IPv6)
- Novell Internetwork Packet Exchange (IPX)
- AppleTalk
- Connectionless Network Service (CLNS/DECNet)

Nejznámějším protokolem a také protokolem kterým se budeme zabývat je protokol IP ve verzi 4.
Jeho základní charakteristikou jsou tři termíny: Nespojovaný (*Connectionless*), Nezávislý na médiu (*Media Independent*) a s nejlepším úsílím (*Best Effort*). Tyto termíny nebo slovní spojení charakterizují podstatu protokolu. Nespojovaný v praxi znamená, že před samotným vysláním paketu není navazováno žádné spojení mezi zdrojovým a cílovým zařízením.

Pojem k zapamatování: nespolehlivost IP protokolu

IP protokol nemá mechanismus kontroly, zdali paket dorazil do místa určení. Fakt, že adresát obdržel data, musí zajistit jiný protokol, v tomto případě protokol TCP transportní vrstvy.

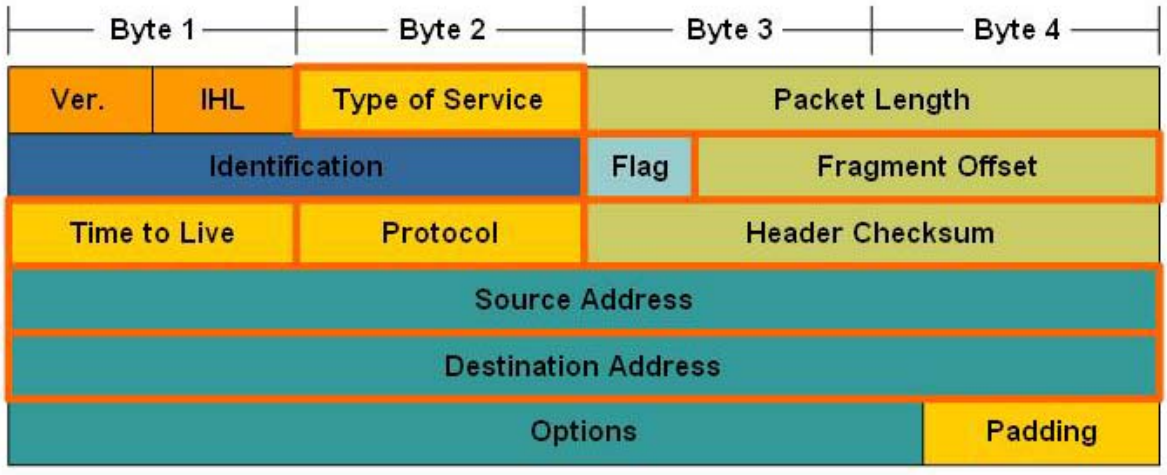
Z toho vyplývá, že zdrojové zařízení neví a pro svůj mechanismus odeslání ani nepotřebuje vědět, zdali je cílové zařízení ještě přítomno v síti, zdali paket dorazil do cíle a zdali jej cílové zařízení umí přečíst. Naopak cílové zařízení neví, ve kterých okamžicích má očekávat příjem paketů. Tento nespojovaný charakter má celou řadu výhod včetně menší zátěže sítě ale také nevýhody, jako např. žádný mechanismus potvrzení příjmu paketů. To musí zajistit protokoly na jiných vrstvách, protože na síťové vrstvě k tomu není dostatečný prostor. S tímto termínem úzce souvisí pojem „S nejlepším úsilím“, routery tak hledají nejlepší cestu pro každý vyslaný paket. Protože jedna zpráva od zdrojového zařízení k cílovému může mít mnoho tisícovek paketů, může se síťová cesta pro různé pakety lišit. Stav a zatížení sítě může být různé, a protože routery neustále hledají pro pakety tu nejlepší cestu, pakety tak mohou dojít zcela v jiném pořadí než byly vyslány.



Obr. 11.2 Hledání nejlepší síťové cesty

Třetí charakteristická vlastnost nezávislost na médiu znamená fakt, že IP protokol neřeší fyzickou vrstvu mezi komunikujícími zařízeními. Nejsou žádná omezení pro IP protokol, která by znemožňovala určitému médiu IP paket přenést.

IP paket je tedy zabalený segment transportní vrstvy opatřený IP hlavičkou. Samotný segment se svými daty zůstává po celou dobu směrování nedotčen. Síťová zařízení podrobně zkoumají obsah IP hlavičky, který je během cesty upravován. Jednotlivé položky hlavičky jsou znázorněny na obrázku 11.3. Již teď známe dvě nejdůležitější z nich: zdrojová a cílová adresa, pojďme si vysvětlit, co znamenají následující položky.



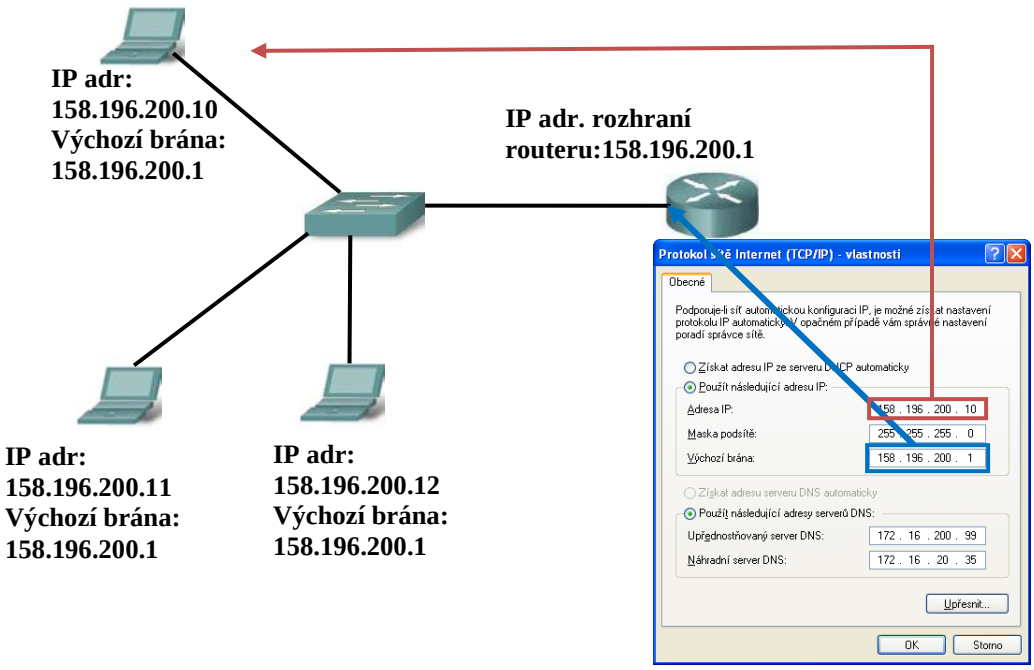
Obr. 11.3 Formát IP hlavičky protokolu

Jednotlivé položky hlavičky IP paketu:

- Verze: verze síťového protokolu IPv = 4.
- IHL: délka hlavičky v 32bitových slovech (4B) IHL= 5 znamená délku 20B.
- Type of Service: osmibitové číslo udávající prioritu paketu.
- Packet Length: délka celého paketu včetně hlavičky a dat.
- Identification, Flag, Fragment Offset: informační bajty identifikující fragmentaci paketu. V některých případech je paket pro médium příliš dlouhý a jeho menší délku musí zajistit právě fragmentace.
- Time to Live: životnost paketu (osmibitové číslo), dekrementaci provádějí routery.
- Protocol: identifikace protokolu vyšší vrstvy:01 ICMP, 06 TCP, 17 UDP.
- Header Checksum: kontrolní součet, pokud neodpovídá, paket je poškozen a bude zničen.

11.3 Default Gateway – brána do venkovní sítě

Tento anglický termín se překládá jako výchozí brána. Termín brána odpovídá opravdovému vstupu do venkovní sítě. V případě, že hostitelský počítač vyšle paket s IP adresou mimo rozsah své sítě, vstupuje do komunikace výchozí brána sítě, která je na směrovači. Tento směrovač rozhodne o síťové cestě tohoto paketu podle své směrovací tabulky (viz následující kapitola).
Na obrázku 11.4 vidíme tři počítače připojené na switch. IP adresy počítačů odpovídají jedné počítačové síti 158.196.200.0 /24 s výchozí bránou do venkovní sítě 158.196.200.1. Adresa výchozí brány je pro tyto počítače stejná a musí být nakonfigurována na každém počítači, který chce komunikovat vně svoji sít.



Obr. 11.4 Nastavení IP adresy počítače a výchozí brány

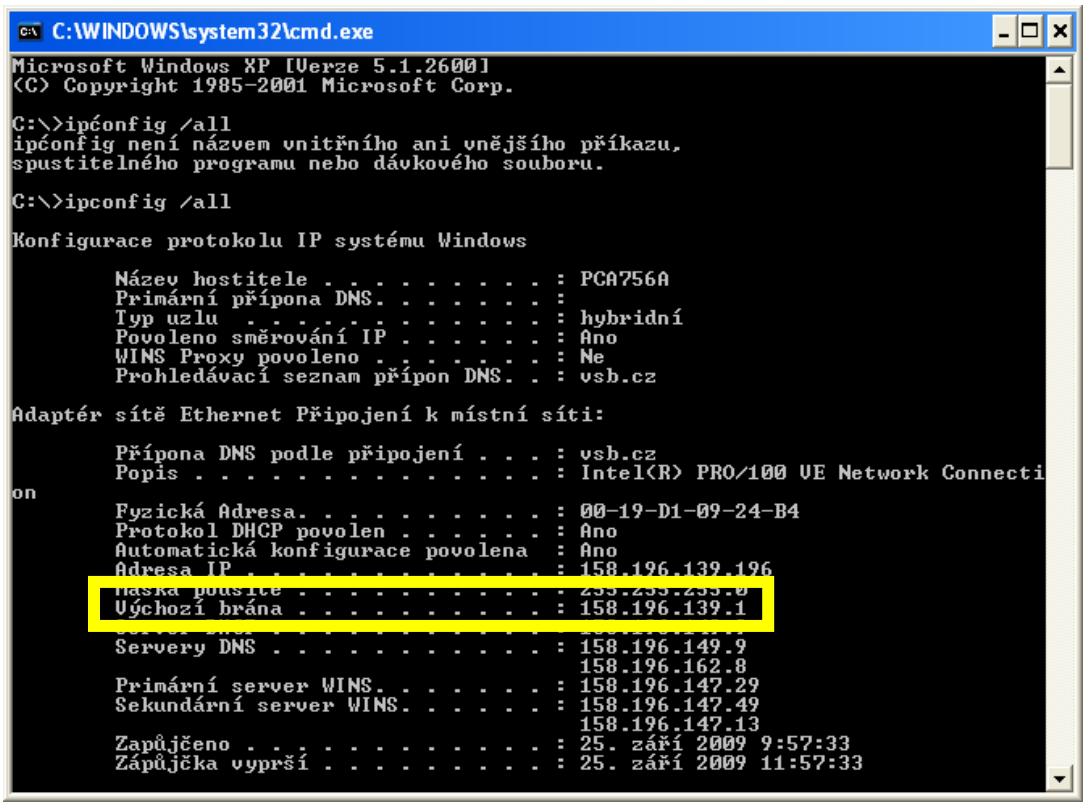
Pojem k zapamatování: Nastavení výchozí brány

Výchozí brána se konfiguruje na hostitelském PC. Pokud tato brána není nastavena, počítač může komunikovat pouze uvnitř vlastní sítě.

IP adresa výchozí brány musí vždy být ve stejné síti jako IP adresa počítače. Tomu musí samozřejmě odpovídat správná síťová maska.

Korespondenční úkol 11.1 – zjištění IP adresy výchozí brány

Příkazem *ipconfig /all* v příkazovém řádku zjistěte IP adresu výchozí brány vašeho počítače do sítě.



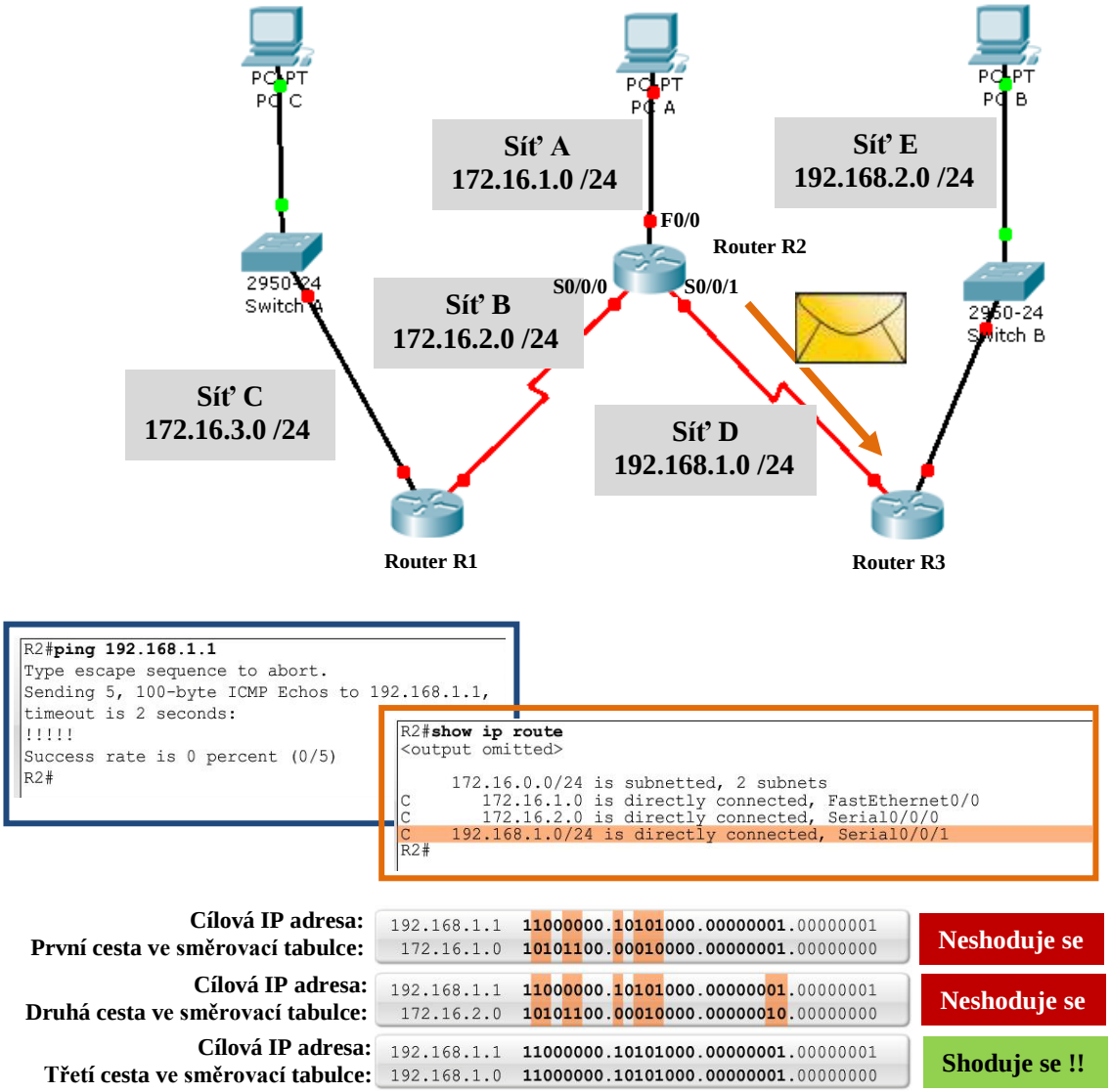
11.4 Síťová cesta a směrovací tabulka

Router potřebuje mít pro své rozhodnutí kam vyslat paket nadefinovanou síťovou cestu. Pokud cesta do sítě odpovídá IP adrese v paketu, je tento paket vyslán touto cestou. Pokud je cílová adresa v paketu z přímo připojené sítě na routeru, vyšle router paket přímo na cílové zařízení. Pokud pro paket router cestu nenajde, je paket nemilosrdně zničen. Na routeru může být také nadefinována speciální defaultní síťová cesta pro pakety, které neodpovídají žádné ze síťových cest, které router zná. Síťové cesty jsou ve své podstatě IP adresy sítí, do kterých zná router cestu. Tyto IP adresy jsou uloženy ve směrovací tabulce routeru.

Směrovací tabulka je vidět na obrázku 11.5. Obsahuje IP adresu sítě, rozhraní routeru, přes které je síť dostupná a také metriku sítě. Ta rozhoduje o prioritě při vyslání paketu, v příadě že budeme mít více možností, kudy poslat paket. V případě směrování se tedy router chová tak jak je to naznačeno na obrázku 11.5. Cílovou adresu porovná se svými cestami ve směrovací tabulce sekvenčně shora dolů. Nejprve cílovou adresu porovná s prvním záznam tabulky a pak následuje porovnání s dalšími záznamy. IP adresy jsou samozřejmě zpracovány binárně a porovnávány jsou jednotlivé pozice bitů. Pokud dojde ke shodě je nalezena cesta pro paket, pokud ke shodě nedojde paket je zahozen.

Pojem k zapamatování: Směrovací tabulka

Směrovací tabulka je tabulka IP adres sítí, do kterých router může vyslat paket, protože zná síťovou cestu. Paket je vyslán korespondujícím rozhraním routeru na svou další cestu.



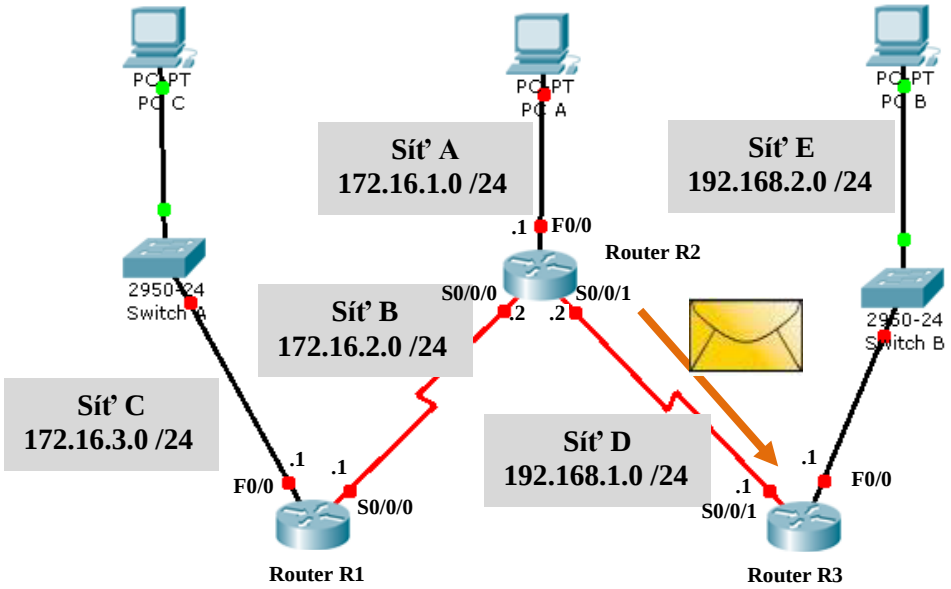
Obr. 11.5 Princip nalezení shody ve směrovací tabulce

11.5 Statické a dynamické směrování

Router se tedy rozhoduje o tom, kam vyšle paket na základě směrovací tabulky. Pokud tedy informace ve směrovací tabulce nejsou aktuální, může dojít při směrování k chybám, paket může cestovat jinou trasou nebo dokonce nemusí dorazit do cíle vůbec. Je tedy jasné, že je v zájmu administrátora sítě, aby směrovací tabulka vždy odpovídala aktuálnímu stavu sítě.
Informace o dostupných síťových cestách mohou být nastaveny manuálně administrátorem nebo naučeny dynamickým způsobem za pomocí směrovacích protokolů. Záleží na topologii sítě, jaké konfigurace pro síťové cesty jsou nejvhodnější, většinou kombinací obou typů – statických i dynamických. Je zřejmé, že sousední segment sítě mohu nastavit staticky, za předpokladu, že vím, že tuto cestu již nikdy nebudu měnit, a samozřejmě že pro vzdálené sítě použiji směrovací protokol, protože o mnohých sítích nemám a ani nemohu mít představu.

Když nakonfigurujeme interface routeru a připojíme na něj naši síť, tento interface se stává výchozí branou sítě a do směrovací tabulky je síť přidána automaticky jako přímo připojená - *"directly connected"*. Nejbližší sítě, o kterých máme představu, že je vhodné je nakonfigurovat staticky, můžeme do směrovací tabulky přidat manuálně.

Na obrázku 11.6 vidíme příkaz na routeru R1, kterým do směrovací tabulky přidáváme cestu do sítě.




```
R1(config)#ip route 192.168.1.0 255.255.255.0 172.16.2.2
R1(config)#ip route 192.168.2.0 255.255.255.0 172.16.2.2
R1(config)#end
R1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 3 subnets
S       172.16.1.0 [1/0] via 172.16.2.2
C       172.16.2.0 is directly connected, Serial0/0/0
C       172.16.3.0 is directly connected, FastEthernet0/0
S    192.168.1.0/24 [1/0] via 172.16.2.2
S    192.168.2.0/24 [1/0] via 172.16.2.2
```

Obr. 11.6 Definice statických cest na routeru

Obecný tvar příkazu by byl po klíčových slovech *ip route* napsat IP adresu a masku sítě, kterou chceme staticky připojit a IP adresu rozhraní nejbližšího routeru, který je přímo k našemu routeru připojen. Výsledkem dvou příkazů na obrázku je fakt, že router R1 bude znát cestu do sítí D a E. V praxi to znamená, že počítač v síti C již může komunikovat s počítačem v síti E, neboť router R1 již zná cestu do této sítě.

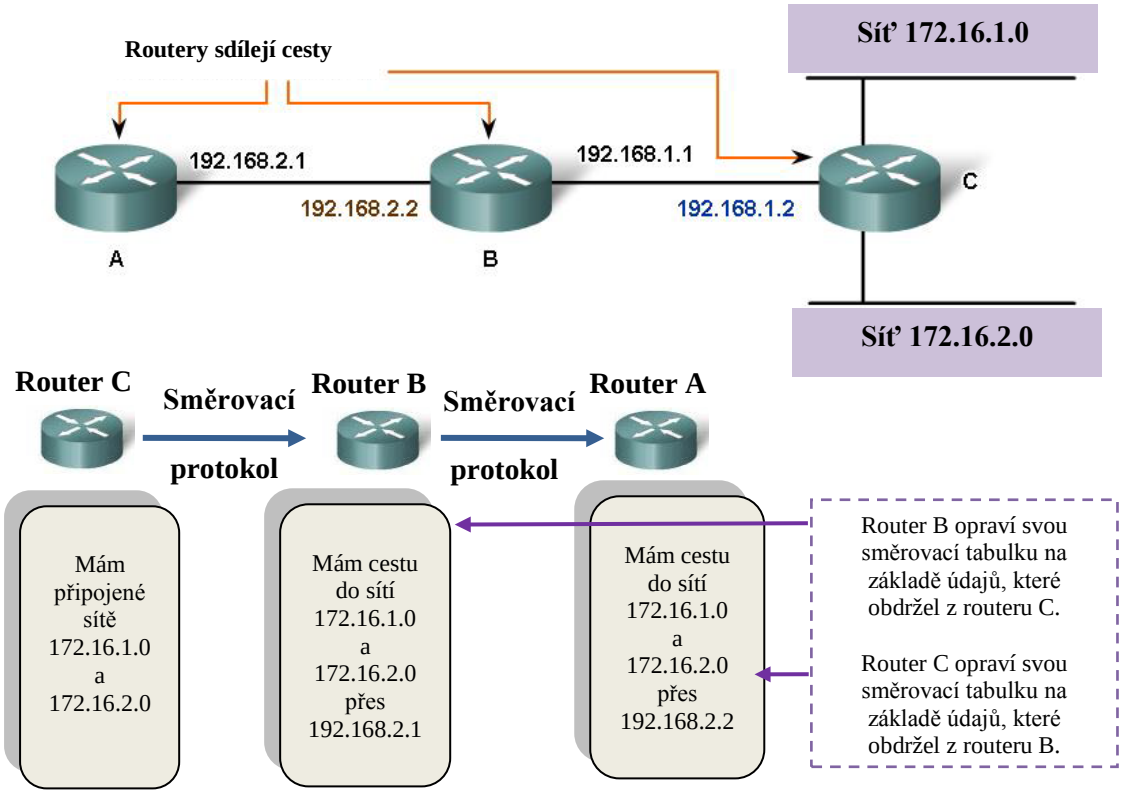
Pojem k zapamatování: Statická cesta

Statická cesta je konfigurovaná manuálně přímo na routeru. Její výhodou je vysoká priorita při výběru cesty při směrování při shodě více cest k cíli, avšak nevýhodou je fakt, že při změně topologie sítě zůstane ve směrovací tabulce špatný záznam, dokud ji administrátor sítě neupraví, router se bude stále řídit podle staré topologie sítě.

Na obrázku 11.6 tedy vidíme směrovací tabulku routeru R1. Směrovací tabulka v tomto případě obsahuje dvě přímo připojitelné cesty a tři statické, které přidal manuálně administrátor. Při výpisu směrovací tabulky si všimněte kódového označení cest: C – *directly connected*, S – *static*, atd. Ostatní písmenka jsou zkratkou pro cesty přidané dynamickým směrovacím protokolem.

Pro větší topologie sítě je nutno využít služeb směrovacích protokolů, protože není v silách administrator zadávat všechny cesty manuálně. Mezi nejznámější směrovací protokoly patří:

- RIP (*Routing Information Protocol*)
- EIGRP (*Enhanced Interior Gateway Protocol*)
- OSPF (*Open Shortest Path First*)



Obr. 11.7 Výměna směrovacích informací pomocí protokolů

Podstatou práce směrovacího protokolu je udržování aktuálních směrovacích tabulek na routerech, které jsou pod stejnou správou směrovacího protokolu. Tyto routery si prostřednictvím protokolu vyměňují své aktuální údaje ze směrovacích tabulek. Pokud se na nějakém routeru změní topologie sítě, router upraví svou směrovací tabulku a informuje o tom ostatní routery. Zjednodušeným příkladem by se tento process předávání informací dal vyjádřit obrázkem 11.7.

Pojem k zapamatování: Dynamický směrovací protokol

Dynamický směrovací protokol zajistí automatickou opravu směrovacích tabulek při změně topologie sítě. Na rozdíl od statických cest se vše děje dynamicky bez zásahu administrátora, avšak tento proces vyžaduje oproti statickým konfiguracím určitou zátěž sítě.

Ke kapitole 11 je přiřazena demonstrační animace č.10 a č. 11

Animace č. 10 obsahuje ukázku konfigurace statických síťových cest na routeru. PC z jedné konkrétní sítě tak bude moci komunikovat s PC z jiné sítě, což by bez nastavení síťové cesty na routeru nebylo možné. Ukázka konfigurace síťové cesty bude provedena v příkazovém řádku routeru, ověření komunikace mezi počítači je možno provést příkazem Ping. Rozdílem oproti předchozí animaci bude v animaci č. 11 ukázka konfigurace

dynamických síťových cest na routeru. PC z jedné konkrétní sítě tak bude moci komunikovat s PC z jiné sítě. Konfigurace síťových cest proběhne za podpory směrovacího protokolu RIP. Ukázka konfigurace síťové cesty bude provedena v příkazovém řádku routeru.

Ke kapitole 11 je přiřazena demonstrační animace č.12, č.13 a č. 14

Animace č. 11 obsahuje ukázku konfigurace dynamické cesty pomocí směrovacího protokolu OSPF. Animace č. 12 ukazuje rozdíl mezi směrovacími tabulkami resp. jejich cestami, které jsou vytvořeny pomocí různého směrovacího protokolu. Animace č. 14 obsahuje ukázku srovnání konfigurace dynamických síťových cest pomocí třech směrovacích protokolů RIP, EIGRP a OSPF, přičemž je demonstrován postup při výběru cesty dle priority směrovacích protokolů.

Shrnutí kapitoly

Síťová vrstva je třetí vrstvou OSI modelu a jejím hlavním úkolem je zajistit výměnu dat mezi jednotlivými uživateli sítě. Zajištění správného chodu této komunikace se skládá ze čtyř základních procesů: Adresování, enkapsulace, směrování, dekapsulace. Síťová vrstva zajišťuje proces zabalení segmentu z nadřazené transportní vrstvy. K tomuto segmentu je přidána IP hlavička, proces enkapsulace tak vytvoří paket síťové vrstvy. IP hlavička obsahuje zdrojovou a cílovou IP adresu komunikačního procesu. Po ukončení enkapsulačního procesu je paket předán nižší vrstvě síťového modelu. Směrování se provádí pomocí zprostředkujících zařízení, kterým říkáme routery.

Nejznámějším síťovým protokolem protokol IP ve verzi 4. Jeho základní charakteristikou jsou tři termíny: Nespojovaný (*Connectionless*), Nezávislý na médiu (*Media Independent*) a s nejlepším úsilím (*Best Effort*). IP protokol nemá mechanismus kontroly, zdali paket dorazil do místa určení. Fakt, že adresát obdržel data, musí zajistit jiný protokol, v tomto případě protokol TCP transportní vrstvy.

Výchozí brána zajišťuje vstup do venkovní sítě. V případě, že hostitelský počítač vyšle paket s IP adresou mimo rozsah své sítě, vstupuje do komunikace výchozí brána sítě, která je na směrovači. Tento směrovač rozhodne o síťové cestě tohoto paketu podle své směrovací tabulky. IP adresa výchozí brány musí vždy být ve stejné síti jako IP adresa počítače.

Směrovací tabulka je tabulka IP adres sítí, do kterých router může vyslat paket, protože zná síťovou cestu. Paket je vyslán korespondujícím rozhraním routeru na svou další cestu. V případě směrování se tedy router chová tak, že cílovou adresu porovná se svými cestami ve směrovací tabulce sekvenčně shora dolů. Nejprve cílovou adresu porovná s prvním záznam tabulky a pak následuje porovnání s dalšími záznamy. IP adresy jsou zpracovány binárně a porovnávány jsou jednotlivé pozice bitů. Pokud dojde ke shodě je nalezena cesta pro paket, pokud ke shodě nedojde paket je zahozen.

Informace o dostupných síťových cestách mohou být nastaveny manuálně administrátorem nebo naučeny dynamickým způsobem za pomocí směrovacích protokolů. Statická cesta je konfigurovaná administrátorem manuálně přímo na routeru. Její výhodou je vysoká priorita při výběru cesty při směrování při shodě více cest k cíli. Pokud však dojde ke změně topologie sítě, ve směrovací tabulce bude špatný záznam, dokud ji administrátor sítě neupraví a router se bude stále řídit podle staré topologie sítě. Naopak dynamický směrovací protokol zajistí automatickou opravu směrovacích tabulek při změně topologie sítě. Na rozdíl od statických cest se vše děje automaticky bez zásahu administrátora, avšak tento proces vyžaduje oproti statickým konfiguracím určitou zátěž sítě.

Úkol k řešení 11.1 – Vytvoření konfigurace sítě

V aplikaci Packet Tracer vytvořte topologii sítě dle obrázku 11.6. Konkrétním zařízením a rozhraním přidělte IP adresy.

Pro IP adresu rozhraní routeru vždy volte první adresu z uvedeného rozsahu. Spojení mezi dvěma routery bude realizováno sériově, v tomto případě budou IP adresy první a druhá adresa z uvedeného rozsahu. Všechny sítě mají masku /24.

Úkol k řešení 11.2 – Vytvoření směrovacích tabulek

Pro topologii z předchozího příkladu vytvořte statické cesty na routerech R1,R2,R3, tak aby bylo možno z jednotlivých PC1, PC2 a PC3 komunikovat mezi sebou. Otestuje výsledek příkazem *ping* mezi jednotlivými PC.

Kontrolní otázka 11.1

Co má na starost síťová vrstva OSI modelu?

Kontrolní otázka 11.2

Jakým způsobem a na kterých zařízeních se provádí enkapsulace?

Kontrolní otázka 11.3

Co znamená termín nespolehlivost IP protokolu?

Kontrolní otázka 11.4

Kterými termíny charakterizujeme IP protokol?

Kontrolní otázka 11.5

Vyjmenujte důležité součásti hlavičky IP protokolu.

Kontrolní otázka 11.6

Co je to výchozí brána?

Kontrolní otázka 11.7

Kde se nastavuje výchozí brána?

Kontrolní otázka 11.8

Co je obsahem směrovací tabulky?

Kontrolní otázka 11.9

Jakým způsobem se postupuje na routeru při směrování?

Kontrolní otázka 11.10

Jaké další údaje obsahuje směrovací tabulka?

Kontrolní otázka 11.11

Jak se konfiguruje statická cesta?

Kontrolní otázka 11.12

Jaký je rozdíl mezi staticky nadefinovanou a dynamicky naučenou cestou?

12. FYZICKÉ PROPOJENÍ, ETHERNET, MAC ADRESY

Čas ke studiu: 1,5 hodiny

Cíl Po prostudování této kapitoly budete umět

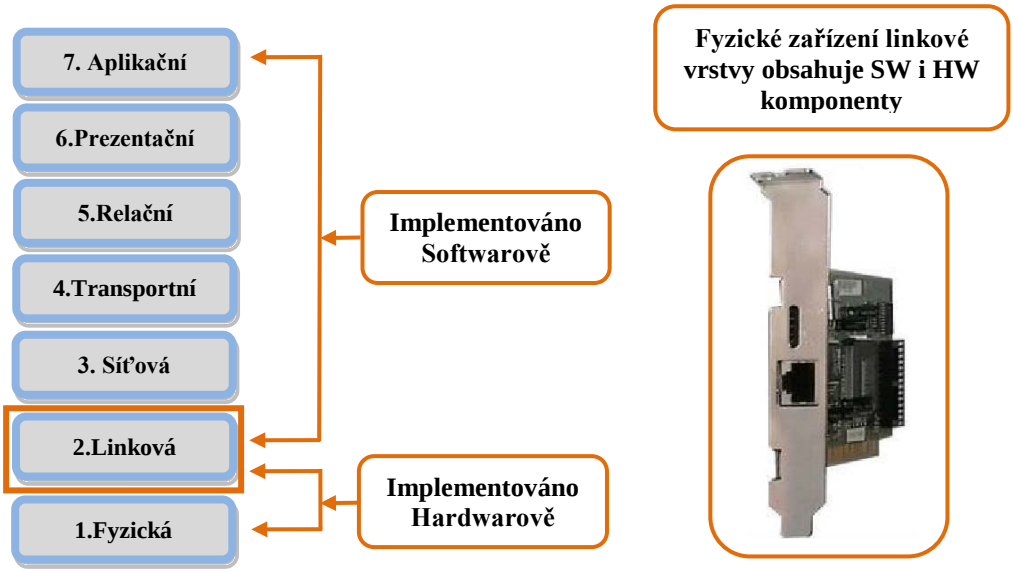
- určit úlohu linkové vrstvy
- popsat protokoly druhé vrstvy síťového modelu
- popsat funkce fyzické vrstvy
- popsat ethernet a jeho implementace
- charakterizovat podstatu MAC adresy
- správně propojovat koncová a síťová zařízení

Výklad

Závěrečná kapitola se věnuje nejspodnějším vrstvám síťového modelu – linkové a fyzické vrstvě. Na těchto vrstvách se již nepracuje s IP adresou, ale s ethernetovou MAC adresou, kterou si podrobněji popíšeme. V úplném závěru tohoto kurzu si definujeme přesná pravidla pro propojování koncových a síťových zařízení.

12.1 Úloha linkové vrstvy

Linková vrstva je velice důležitá, poskytuje mechanismus vyšším vrstvám pro přístup ke komunikačnímu médiu. To se děje prostřednictvím zapouzdření do rámců. Tyto rámce posílá linková vrstva po komunikačním médiu, zajišťuje adresování rámců, označuje jejich začátek a konec a také detekci kolizí a chyb. Linková vrstva na komunikačním médiu řídí komunikaci mezi jednotlivými uzly, obsahuje kontrolní informace o tom, kdo s kým komunikuje, kdy komunikace začíná a končí, který uzel bude komunikovat poté a jaké chyby se mohou vyskytovat při komunikaci.



Obr. 12.1 Linková vstva je implementována softwarově i hardwarově

Pojem k zapamatování: Propojení vyšších vrstev s komunikačním médiem

Linkovou vrstvu můžeme chápat jako propojení softwarových procesů horních vrstev s fyzickou hardwarovou vrstvou pod nimi.

Z výše uvedeného vyplývá, že linková vrstva by se dala rozdělit na dvě podvrstvy. Horní podvrstva (angl. *sublayer*) definuje softwarový proces, který zajistí služby protokolům síťové vrstvy a spodní podvrstvu, která definuje procesy přístupu k médiu zajištěné hardwarovou implementací. Ve skutečnosti existuje detailní specifikace úlohy těchto dvou podvrstev s názvem LLC (*Logical Link Layer*) a MAC (*Media Access Control*). Spodní vrstva MAC definuje řízení dat na sběrnici různých topologií a metody detekcí kolizí CSMA/CD a CSMA/CA.

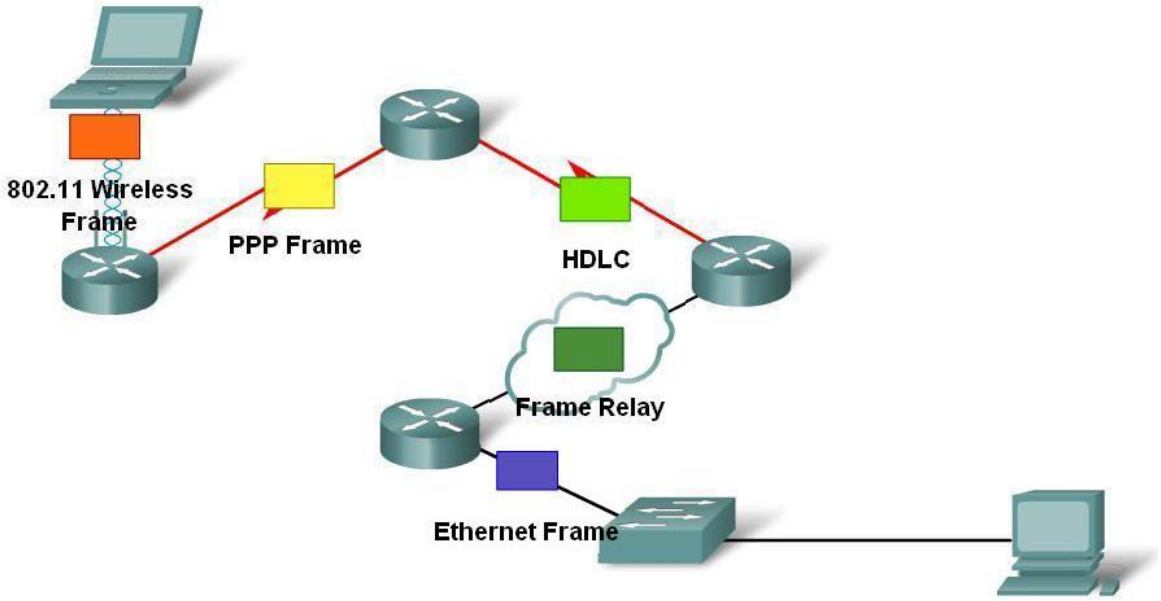
12.2 Rámec linkové vrstvy

Rámec linkové vrstvy obsahuje zabalený paket vyšší vrstvy doplněný o hlavičku a také zakončení. Hlavička je samozřejmě nezbytná pro adresaci, musí obsahovat adresu zdroje a cíle, nikoliv však ve formátu IP adresy. Další nezbytnou součástí hlavičky je určení začátku každého rámce, určení protokolu vyšší vrstvy, který převezme rozbalený paket uvnitř rámce a další informace, jako priorita a řízení paketu, tyto údaje se však liší v jednotlivých protokolech linkové vrstvy a tak budeme za strukturu rámce považovat zobrazení na obr. 12.2

Header			Data	Trailer	
Start Frame	Adresa	Typ/Délka		FCS	Stop Frame

Obr. 12.2 Rámec linkové vrstvy

Na konci každého rámce je takzvané zakončení, to určuje pole Stop Frame indikující konec rámce a FCS, což je kontrolní součet, který sebou každý rámec nese. Tento kontrolní součet vložil vysílající uzel na základě obsahu rámce. Přijímající uzel obdrží rámec a rovněž vytvoří kontrolní součet na základě přijatých bitů, a pokud vypočtený součet nesouhlasí s obsahem pole FCS, je jasné že došlo k chybnému přenosu.
Jak už bylo řečeno, přesná struktura rámce linkové vrstvy závisí na implementovaném protokolu linkové vrstvy. Ten je z části závislý na logické topologii sítě a z části na implementaci fyzického media. Pro zjednodušení bezdrátová WIFI síť bude mít jistě jiný formát rámce než ethernetový protokol.



Obr. 12.3 Příklady protokolů druhé vrstvy síťového modelu

Vyjmenujeme-li nejznámější protokoly linkové vrstvy, musíme zmínit:

- Ethernet protocol,
- PPP – Point to Point Protocol,
- HDLC – High Level Data Link Control protokol,
- Frame Relay protocol,
- ATM – Asynchronous Transfer Mode protocol.

Některé protokoly jsou přizpůsobeny technologii LAN s mnoha uzly na kratší vzdálenost, jiné protokoly pro sítě WAN s méně uzly. Typickým protokolem WAN je PPP protokol, jež má pouze dva uzly- vysílač a přijímač, a proto není nutné, aby rámec obsahoval zdrojovou a cílovou adresu. Naopak protokol 802.11 Wireless LAN obsahuje ve své struktuře několik adres, neboť potřebuje detekovat více bezdrátových uzlů. Pro ucelení představy o nejpoužívanějším protokolu Ethernet uvádíme jeho strukturu rámce, včetně velikosti jednotlivých položek. Z obrázku vyplývá, že Ethernet protokol dokáže v jednom rámci pojmout až 1,5 KB uživatelských dat. Z tohoto příkladu si patrně každý udělá obrázek, kolik rámců musí být vysláno při různě velikém objemu dat.

Preambule	Cíl	Zdroj	Typ	Data	FCS
8 bytů	6 bytů	6 bytů	2 byty	46 – 1500 bytů	4 byty

Obr. 12.4 Rámec protokolu Ethernet

- **Preambule** se používá pro synchronizaci rámců, obsahuje ukončovací značku při časování.
- **Cílová adresa** je 48 bitová MAC adresa příjemce.
- **Zdrojová adresa** je 48 bitová MAC adresa odesílatele.
- **Typ** určuje protokol vyšší vrstvy, který obdrží data po rozbalení rámce.
- **Data** jsou v tomto případě PDU vyšší vrstvy, obvykle IPv4 paket.
- **FCS (Frame Check Sequence)** je kontrolní součet rámce pro detekci chyb.

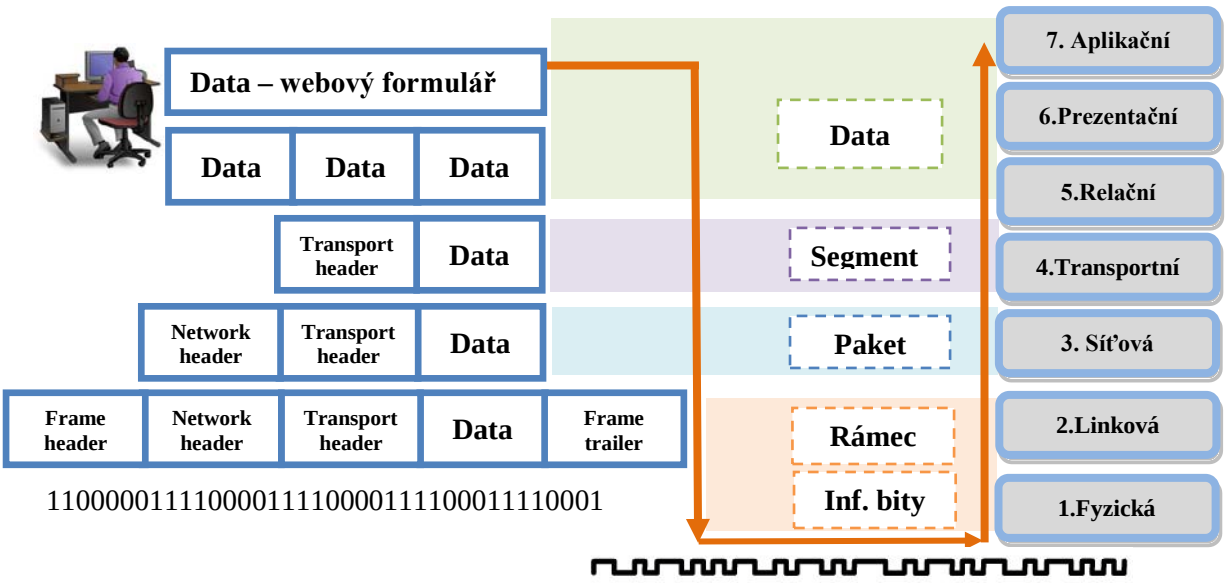
12.3 Úloha fyzické vrstvy

Fyzická vrstva ISO/OSI modelu zajišťuje transport rámce linkové vrstvy na úrovni bitů. Fyzická vrstva implementuje 4 nejdůležitější součásti:

- Fyzická media a k nim asociované konektory.
- Reprezentaci posloupnosti bitů na konkrétním médiu.
- Kódování dat a kontrolních informací.
- Zajištění vysílání a přijímání na síťových zařízeních.

Reprezentace posloupnosti bitů závisí na konkrétním typu přenosového média, tím je obecně měděný kabel, optická linka nebo bezdrátové médium, čili vzduch. Standardizací fyzické vrstvy se zabývají organizace jako ISO, IEEE, ANSI, ITU a EIA/TIA. Tyto organizace stanovují hardwarové standardy, které se týkají fyzikálních a elektrických vlastností přenosového media, mechanické vlastnosti médií, konektory, materiály, počty a definice pinů konektorů, definice kontrolních informací a enkódování dat, to znamená bitová reprezentace konkrétního signálu.

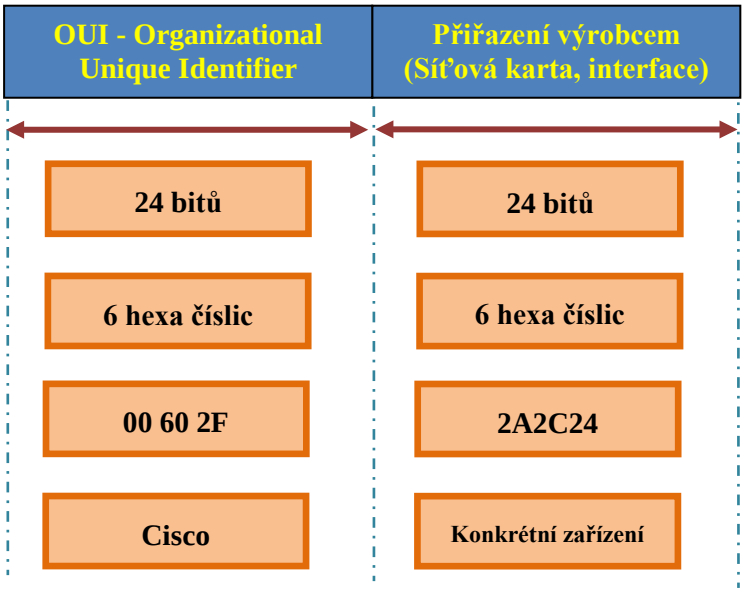
V souvislosti s touto problematikou definujeme dva angl. pojmy tzv. *Encoding* a *Signaling*. Encoding je převod určitého proudu datových bitů na nějaký předefinovaný kód podle bitové šablony, které rozumí vysílající i přijímající uzel. Signaling je pak fyzická reprezentace těchto nul a jedniček pomocí nějaké signální metody, tedy generováním elektrického signálu nebo optických pulzů. Signální metody se mohou určovat amplitudou, frekvencí či fází. Příkladem signální metody je NRZ či NRZI.



Obr. 12.5 Postup transformace dat až směrem k bitům fyzické vrstvy

12.4 Ethernetová MAC adresa

Mac adresa neboli fyzická adresa je jednoznačným identifikátorem síťového interface každého zařízení. Cílová MAC adresa je rozhodujícím faktorem zdali cílové zařízení rámec přijme nebo zahodí. Formát MAC adresy je 48 bitový rozdělen na dvě poloviny. První polovinu přiřazuje výrobců standard IEEE, druhou polovinou rozlišují své síťové karty sami výrobci.



Obr. 12.6 Příklad formátu MAC adresy

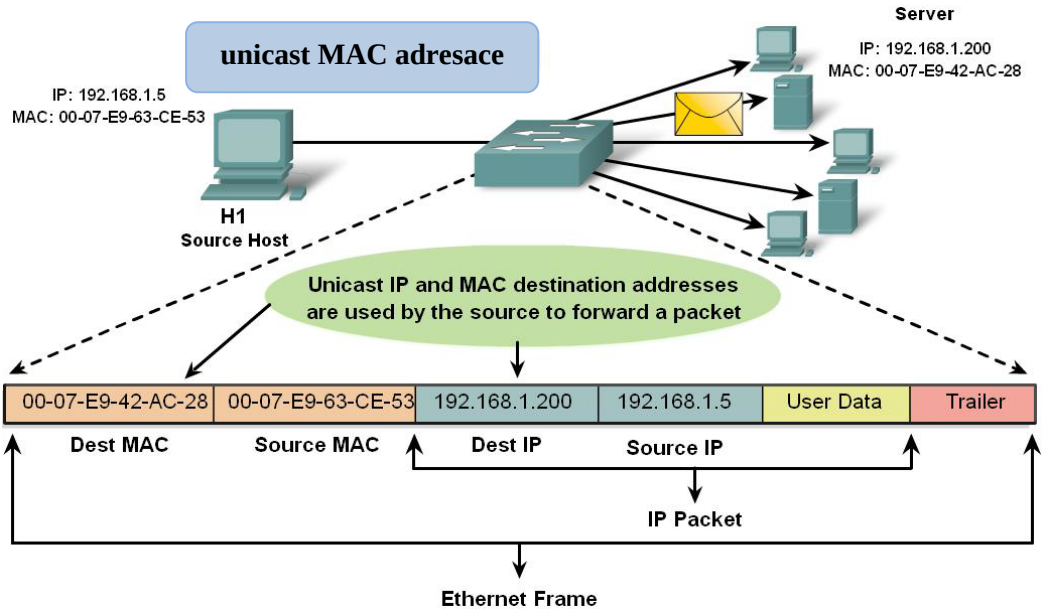
MAC adresu ve vašem počítači zjistíte známým příkazem *ipconfig /all*. MAC adresa uvedena na obrázku 12.6 může být zobrazena ve třech formách zápisu: 00-60-2F-2A-2C-24 nebo 00:60:2F:2A:2C:24 a nebo 00.60.2F.2A.2C.24.

Stejně tak jako v síťové vrstvě existují tři typy adresace: unicast, broadcast, multicast, i na úrovni rámců definujeme tento druh komunikace. Broadcastová MAC adresa je FF-FF-FF-FF, multicastovou MAC adresu poznáme tak, že začíná vždy 01-00-5E. Na obrázku 12.7 vidíme ethernetový rámec včetně adresace a zabaleného paketu síťové vrstvy. Zde se jedná o adresaci typu unicast.

Pojem k zapamatování: MAC adresa

MAC adresa je 48 bitový identifikátor zařízení vyjádřený ve formě 12 hexa číslic. První polovina MAC adresy určuje tzv. OUI, neboli výrobce, ten pak přiřadí zbývajících 6 hexa číslic adresy.

Standard IEEE zpřístupnil online vyhledávací nástroj pro zjištění OUI každého zařízení: http://standards.ieee.org/regauth/oui/index.shtml

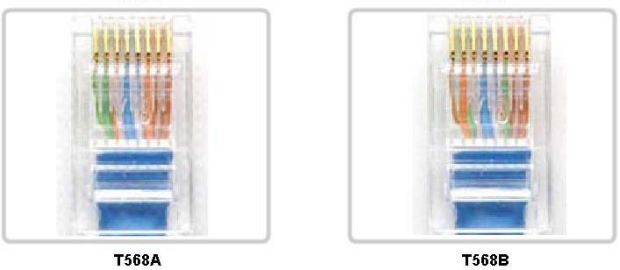


Obr. 12.7 Obsah ethernetového rámce

Uvedený postup je samozřejmě velice zjednodušený. Ve skutečnosti každé síťové zařízení, provádí dekapsulaci obdrženého paketu, samozřejmě ne až do poslední vrstvy, protože ho cílová data nezajímají. Síťové zprostředkující zařízení ale bude dekapsulovat paket na úroveň své vrstvy a opět paket zabalí a pošle k následujícímu síťovému zařízení. Až poslední zařízení, které je cílové rozbaluje paket až po aplikační data.

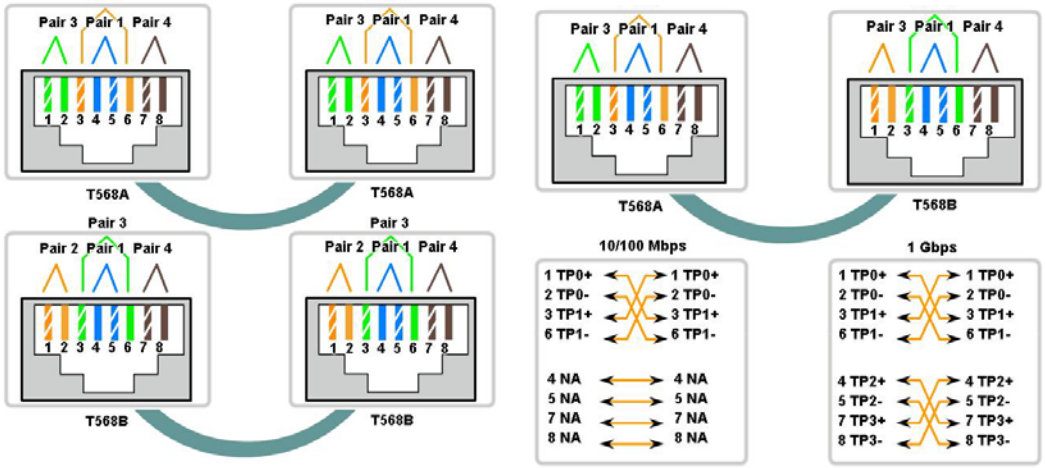
12.5 Propojování zařízení

Pro běžné uživatele nejznámějším propojovacím kabelem je kabel s konektorem RJ-45. Tento kabel většinou propojuje zásuvku připojenou na počítačovou síť se síťovou kartou počítače. Kabel a konektor je specifikován dle standard EIA/TIA (*Electonics Industry Alliance/Telecommunications Industry Association*). Kabel RJ-45 obsahuje osm pinů v tzv. 4 párech, které jsou barevně odlišeny. Existují dvě variant zakončení konektoru, variant T568A a T568B, lišící se přehozením párů pro využití ve volbě kabelu.



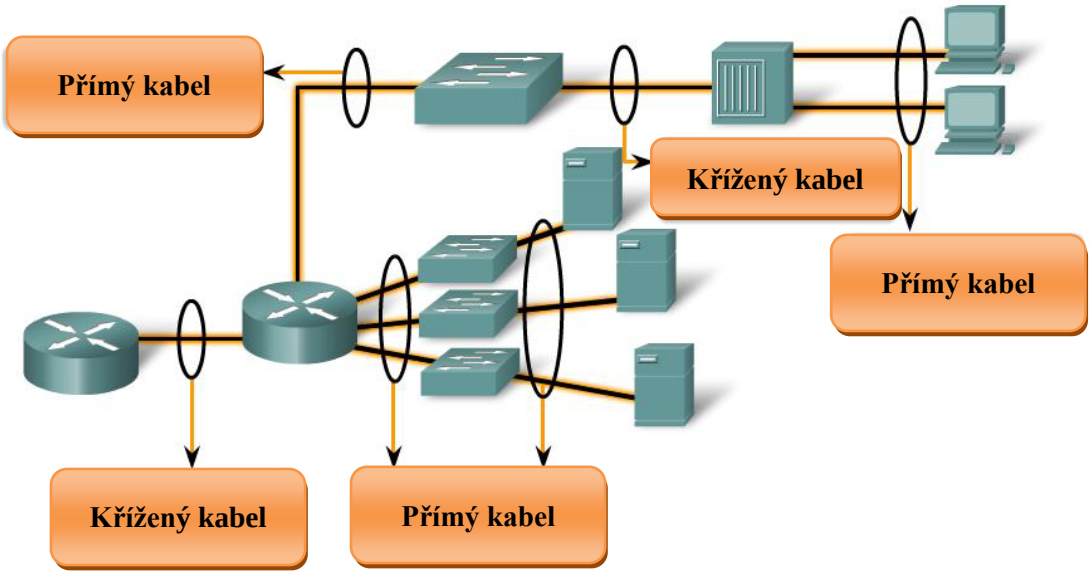
Obr. 12.8 Zakončení LAN kabelu konektorem RJ-45 ve dvou verzích

Dva typy zakončení nám umožní vytvořit dva typy kabelu, přímý a tzv. křížený, kterým často uživatelé propojují dva počítače mezi sebou pro rychlejší přenos dat. Přímý kabel je charakterizován stejným typem zakončení konektorů na obou stranách tedy T568A-T568A nebo T568B- T568B. Křížený kabel má tudíž na jeden straně zakončení T568A a na druhé straně zakončení T568B.



Obr. 12.9 Propojení zařízení přímým a kříženým LAN kabelems RJ-45 konektory

Přímým kabelem zapojujeme počítač do switche nebo routeru a nebo switch k ethernetovém portu routeru. Pro připojení stejných zařízení mezi sebou potřebujeme použít křížový kabel. Volíme ho tedy pro volby počítač – počítač, switch – switch, router – router (myšleno ethernetový port). Křížový kabel použijeme take u propojení switch – hub nebo počítač – router (ethernetový port).



Obr. 12.10 Propojení zařízení pomocí přímého a kříženého kabelu

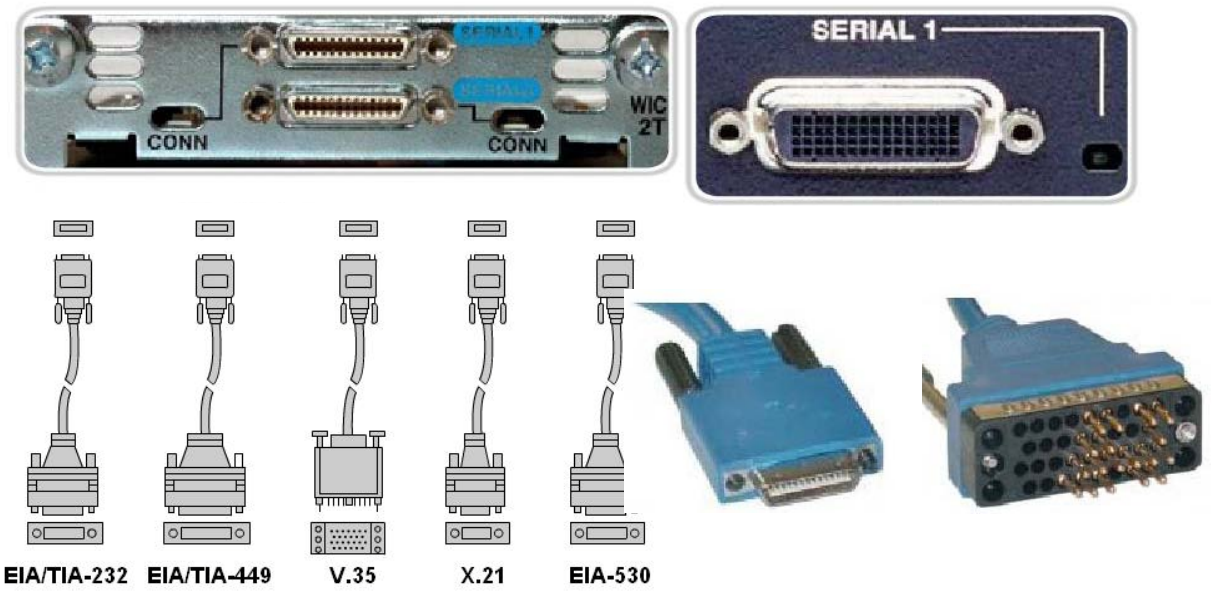
Pojem k zapamatování: Propojování zařízení

Propojování zařízení můžeme zjednodušit tak, že si rozdělíme zařízení na dvě skupiny. V první skupině bude hub a switch a ve druhé počítač a router. Pokud budeme propojovat zařízení, které patří do stejné skupiny, použijeme křížový kabel, pokud propojíme zařízení z první skupiny se zařízením z druhé skupiny, použijeme kabel přímý.

Propojování routeru na dlouhé vzdálenosti pomocí seriového kabelu se opět řídí standardem EIA/TIA. Seriové konektory podle tabulky 12.1 rozdělíme na tři typy, dle použitých typů propojení.

Tab. 12.1 Typy konektorů WAN

Seriové WAN propojení				
Cisco HDLC	PPP	Frame Relay	DSL Modem	Cable Modem
EIA/TIA-232, EIA/TIA-449, X21.V.24, V35, HSSI (High Speed Serial Interface)			RJ-11 telefonní linka	F linka kabelové TV



Obr. 12.11 Seriové WAN rozhraní routeru a WAN konektory

Ke kapitole 12 je přiřazena demonstrační animace č. 12

Animace č. 12 obsahuje ukázku rozdílného chování Hubu a Switche, zatímco hub přeposílá pakety na všechny připojené PC, switch pouze na cílový počítač. Výstupem animace je rovněž ukázka arp tabulky switche a koncového zařízení.

Shrnutí kapitoly

linková vrstva posílá po komunikačním médiu rámce, zajišťuje jejich adresování, označuje jejich začátek a konec a také detekuje kolize a chyby. Linková vrstva na komunikačním médiu řídí komunikaci mezi jednotlivými uzly, obsahuje kontrolní informace o tom, kdo s kým komunikuje, kdy komunikace začíná a končí, který uzel bude komunikovat poté a jaké chyby se mohou vyskytovat při komunikaci.

Linkovou vrstvu můžeme chápat jako propojení softwarových procesů horních vrstev s fyzickou hardwarovou vrstvou pod nimi.

Rámec linkové vrstvy obsahuje zabalený paket vyšší vrstvy doplněný o hlavičku a také zakončení. Hlavička je nezbytná pro adresaci, musí obsahovat adresu zdroje a cíle, nyní však ve formátu MAC adresy. Další nezbytnou součástí hlavičky je určení začátku každého rámce, určení protokolu vyšší vrstvy, který převezme rozbalený paket uvnitř rámce, konec rámce a kontrolní součet. Přesná struktura rámce linkové vrstvy závisí od implementovaného protokolu linkové vrstvy. Ten je z části závislý na logické topologii sítě a z části na implementaci fyzického media. Nejznámější protokoly linkové vrstvy

jsou: Ethernet protokol, PPP – Point to Point Protocol, HDLC – High Level Data Link Control protocol, Frame Relay protocol a ATM – Asynchronous Transfer Mode protocol. Z formátu rámce protokolu Ethernet vyplývá, že protokol dokáže v jednom rámci pojmout až 1,5 KB uživatelských dat.

Fyzická vrstva ISO/OSI modelu zajišťuje transport rámce linkové vrstvy na úrovni bitů. Fyzická vrstva implementuje 4 nejdůležitější součásti: Fyzická media a k nim asociované konektory, reprezentaci posloupnosti bitů na konkrétním médiu, kódování dat a kontrolních informací a zajištění vysílání a přijímání na síťových zařízeních.

Fyzická vrstva reprezentuje signál pomocí metod *encoding* a *signaling*. *Encoding* je převod určitého proudu datových bitů na nějaký předefinovaný kód podle bitové šablony, které rozumí vysílající i přijímající uzel. *Signaling* je pak fyzická reprezentace těchto nul a jedniček pomocí nějaké signální metody, tedy generováním elektrického signálu nebo optických pulzů.

Mac adresa neboli fyzická adresa je jednoznačným identifikátorem síťového interface každého zařízení. Cílová MAC adresa je rozhodujícím faktorem zdali cílové zařízení rámec přijme nebo zahodí. Formát MAC adresy je 48 bitový rozdělen na dvě poloviny. První polovinu přiřazuje výrobců standard IEEE, druhou polovinou rozlišují své síťové karty sami výrobci. MAC adresu ve vašem počítači zjistíte známým příkazem *ipconfig /all*. MAC adresa uvedena na obrázku 12.6 může být zobrazena ve třech formách zápisu: 00-60-2F-2A-2C-24 nebo 00:60:2F:2A:2C:24 a nebo 00.60.2F.2A.2C.24.

Stejně tak jako v síťové vrstvě existují tři typy adresace: unicast, broadcast, multicast, i na úrovni rámců definujeme tento druh komunikace. Broadcastová MAC adresa je FF-FF-FF-FF, multicastovou MAC adresu poznáme tak, že začíná vždy 01-00-5E.

Kabel s konektorem RJ-45 je pro běžné uživatele nejznámnějším propojovacím kabelem. Tento kabel většinou propojuje zásuvku připojenou na počítačovou síť se síťovou kartou počítače. Kabel RJ-45 obsahuje osm pinů v tzv. 4 párech, které jsou barevně odlišeny. Existují dvě variant zakončení konektoru, variant T568A a T568B, lišící se přehozením párů pro využití ve volbě kabelu. Dva typy zakončení nám umožní vytvořit dva typy kabelu, přímý a tzv. křížený, kterým často uživatelé propojují dva počítače mezi sebou pro rychlejší přenos dat. Přímý kabel je charakterizován stejným typem zakončení konektorů na obou stranách tedy T568A-T568A nebo T568B- T568B. Křížený kabel má tudíž na jeden straně zakončení T568A a na druhé straně zakončení T568B. Pro propojování zařízení můžeme vytvořit pravidlo tak, že si rozdělíme zařízení na dvě skupiny. V první skupině bude hub a switch a ve druhé počítač a router. Pokud budeme propojovat zařízení, které patří do stejné skupiny, použijeme křížový kabel, pokud propojíme zařízení z první skupiny se zařízením z druhé skupiny, použijeme kabel přímý.

Úkol k řešení 12.1 – Propojení křížovým kabelem

Propojte vzájemně dvě PC pomocí křížového kabelu. Natavte IP adresy a vyzkoušejte kopírování dat z jednoho PC na druhé.

Úkol k řešení 12.2 – Zjištění výrobce síťové karty z MAC adresy

Zjistěte výrobce vaší MAC adresy tzv. OUI podle online nástroje:

http://standards.ieee.org/regauth/oui/index.shtml

Kontrolní otázka 12.1

Jakým způsobem bychom mohli stručně charakterizovat úlohu linkové vrstvy?

Kontrolní otázka 12.2

Jaké typy protokolů existují na druhé vrstvě síťového modelu a v čem se liší?

Kontrolní otázka 12.3

Jaká je úloha fyzické vrstvy?

Kontrolní otázka 12.4

Jaký je formát ethernetového rámce?

Kontrolní otázka 12.5

Kolik kilobytů uživatelských dat může obsahovat jeden rámec?

Kontrolní otázka 12.6

Co je to MAC adresa a co nám určuje její hodnota?

Kontrolní otázka 12.7

Jak jsou definována pravidla pro propojování síťových a koncových zařízení?

Kontrolní otázka 12.8

Jakým kabelem propojíme server s ethernetovým rozhraním routeru?

Výběr doporučené literatury pro oblast webdesignu a počítačových sítí

Profesionální webdesign

Techniky a vzorová řešení pro XHTML a CSS

Clint Eccher

672 stran, rok vydání 2010
ISBN: 978-80-251-2677-6

333 tipů a triků pro CSS

Martin Domes

272 stran, rok vydání 2009
ISBN: 978-80-251-2360-7

Mistrovství v CSS

Pokročilé techniky pro webové designéry a vývojáře

Jeff Croft, Ian Lloyd, Dan Rubin

416 stran, rok vydání 2007,
ISBN: 978-80-251-1705-7

HTML, XHTML a CSS

Názorný průvodce tvorbou WWW stránek

Elizabeth Castro

440 stran, rok vydání 2007,
ISBN: 978-80-251-1531-2

HTML a CSS

Krok za krokem

Faithe Wempen

328 stran , rok vydání 2007,
ISBN: 978-80-251-1505-3

CSS

Využijte kaskádové styly naplno!

Charles Wyke-Smith

256 stran, rok vydání 2006,
ISBN: 80-251-1297-7

Webdesign

Nenuťte uživatele přemýšlet!, 2. aktualizované vydání

Steve Krug

168 stran, rok vydání 2006,
ISBN: 80-251-1291-8

XML

Krok za krokem, 2. vydání

Michael J. Young

472 stran, rok vydání 2006,
ISBN: 80-251-1070-2

HTML a CSS

Velká kniha řešení

Marianne Hauser, Tobias Hauser, Christian Wenz

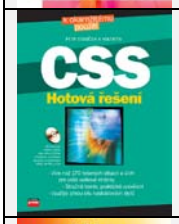
912 stran, rok vydání 2006,
ISBN: 80-251-1117-2

Flexibilní webdesign

Vytváříme přizpůsobitelné a přístupné stránky pomocí XHTML a CSS

Dan Cederholm

232 stran, rok vydání 2006,
ISBN: 80-251-1018-4



CSS

Hotová řešení

Petr Staníček a kolektiv

268 stran, rok vydání 2006,
ISBN: 80-251-1031-1

Vytváříme WWW stránky a spravujeme moderní web site

7. aktualizované vydání

Jiří Hlavenka a kolektiv

368 stran, rok vydání 2005,
ISBN: 80-251-0801-5

CSS kaskádové styly pro webdesignéry

2. vydání

Marek Prokop

288 stran, rok vydání 2005,
ISBN: 80-251-0487-7

Tvorba webů podle standardů

XHTML, CSS, DOM, ECMASkript a dalších

Jeffrey Zeldman

416 stran, rok vydání 2005,
ISBN: 80-251-0347-1

Web design

Kompletní průvodce

Thomas A. Powell

844 stran, rok vydání 2004,
ISBN: 80-722-6949-6

HTML a DHTML

Hotová řešení

Imrich Buranský

272 stran, rok vydání 2003,
ISBN: 80-722-6841-4

XML v kostce

Elliotte Rusty Harold, W. Scott Means

456 stran, rok vydání 2002,
ISBN: 80-7226-712-4

JavaScript pro webové vývojáře

Programujeme profesionálně

Nicholas Z. Zakas

832 stran, rok vydání 2009,
ISBN: 978-80-251-2509-0

JavaScript

Krok za krokem

Steve Suehring

336 stran, rok vydání 2008,
ISBN: 978-80-251-2241-9

JavaScript

Hotová řešení

Petr Václavek

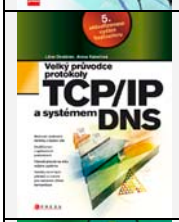
256 stran, rok vydání 2003,
ISBN: 80-7226-854-6

JavaScript

Programujeme internetové aplikace, 2. aktualizované vydání

Rastislav Škultéty

224 stran, rok vydání 2004,
ISBN: 80-251-0144-4



Velký průvodce protokoly TCP/IP a systémem DNS

5. aktualizované vydání bestselleru

Alena Kabelová, Libor Dostálek

488 stran, rok vydání 2008,
ISBN: 978-80-251-2236-5

Moderní komunikační sítě od A do Z

2. aktualizované vydání

Rita Pužmanová

432 stran, rok vydání 2006,
ISBN: 80-251-1278-0

Směrování a přepínání sítí

Autorizovaný výukový průvodce

Wendell Odom, Rus Healy, Naren Mehta

880 stran, rok vydání 2009,
ISBN: 978-80-251-2520-5

Směrování v sítích IP

Mark A. Sportack
368 stran, rok vydání 2004,
ISBN: 80-251-0127-4

Počítačové sítě

Bez předchozích znalostí

Wendell Odom
384 stran, rok vydání 2005,
ISBN: 80-251-0538-5

Počítačové sítě pro začínající správce

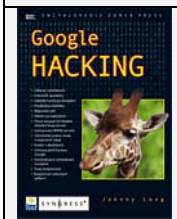
4. aktualizované a rozšířené vydání

Jaroslav Horák, Milan Keršláger
328 stran, rok vydání 2008,
ISBN: 978-80-251-2073-6

Hacking bez tajemství

3. aktualizované vydání

Stuart McClure, Joel Scambray, George Kurtz
632 stran, rok vydání 2004,
ISBN: 80-722-6948-8



Google Hacking

Johny Long
472 stran, rok vydání 2005,
ISBN: 978-80-251-2520-5

Hacking - umění exploitace

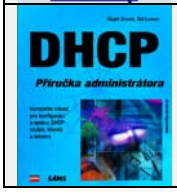
Jon Erickson
544 stran, rok vydání 2009,
ISBN: 978-80-7413-022-9

TCP/IP v kostce

Rita Pužmanová
619 stran, rok vydání 2009,
ISBN: 978-80-7232-388-3

Bezpečnost sítí na maximum

100 tipů a opatření pro okamžité zvýšení bezpečnosti vašeho serveru a sítě
Andrew Lockhart
276 stran, rok vydání 2008,
ISBN: 80-251-0805-8



DHCP Příručka administrátora

Ralph Droms, Ted Lemon
528 stran, rok vydání 2004,
ISBN: 80-251-0130-4

Klíč k řešení – kapitola 1

Odpověď na kontrolní otázku 1.1

Konkurenční prohlížeče jsou založeny na různých aplikačních jádrech. Jádra prohlížečů určují rozdílné vykreslovací schopnosti prohlížečů, některé prvky jsou odlišné i v bezpečnostním přístupu. Nejen těmito vlastnostmi se uvedené prohlížeče liší. V dnešní a budoucí době bude o popularitě prohlížečů rozhodovat nejen bezpečnost, ale také jednotlivé vložené nové moduly, tzv. *plugginy*, které uživatelům nabízejí určitý komfort v oblasti, kterou využívají. Některé prohlížeče, tak přímo komunikují prostřednictvím webových služeb s různými servery, jako například Flickr pro sdílení a správu fotografií, jiné prohlížeče obsahují podporů různých komunikačních programů a messengerů apod. Proto nelze jednoznačně určitý prohlížeč preferovat před jiným a proto také vývojáři webových aplikací musí dbát na to, aby jejich aplikace pod těmito prohlížeči fungovaly. Některé prohlížeče jsou přímo výhodné pro použití v určitých typech operačního systému rozdílného od platformy Windows. Nezanedbatelným faktorem je také rostoucí trend mobilních zařízení s možností připojení internetu. Tyto zařízení vyžadují opět webové prohlížeče na míru operačnímu systému a vybavení zařízení.

Odpověď na kontrolní otázku 1.2

Mezi prohlížeče, které nebyly dosud v této kapitole popsány patří mimo jiné Sea Monkey na jádře Gecko, či Maxthon na jádře shodném s IE.

Obr. 13.1 Další používané prohlížeče

Odpověď na kontrolní otázku 1.3

SGML je standard pro zobrazování textových informací (*Standard Generalized Markup Language*). SGML je univerzální značkovací jazyk dán ISO standardem a umožňuje definovat různé typy značkovacích jazyků podle své syntaxe a pravidel. Značkovací jazyk je to proto, že na rozdíl od programovacích jazyků, kdy určité konstrukci klíčových slov rozumí kompilátor programu, zde pomocí značek se oznámí cíli, jak má daný úsek informací mezi značkami zobrazit. Ze standardu SGML vchází značkovací jazyk HTML a také rozšířený jazyk XML.

Odpověď na kontrolní otázku 1.4

HTML jazyk se vyvíjel od roku 1989. V roce 1994 byla uveřejněna verze 2.0, která již odpovídá standardům SGML. Verze jazyka 3.0 nebyla přijata jako standard pro svou velkou složitost, až v roce 1996 po opravách byla přijatá verze 3.2. V současné době se používá verze 4.01, která byla vydána již v roce 1999. Na specifikaci verze html 5 se neustále pracuje a pravděpodobně bude uvolněna v roce 2012 a standardizována deset let poté v roce 2022. Deset let si nechávají vývojáři na odladění všech chyb ve specifikaci.

Odpověď na kontrolní otázku 1.5

Nepárové elementy jsou např. zalomení řádku
 nebo obrázek <img> a jiné. Tyto elementy nemají párové ukončení, a proto je dle současných standardů ukončujeme lomítkem před samotným ukončením elementu hranatou závorkou:

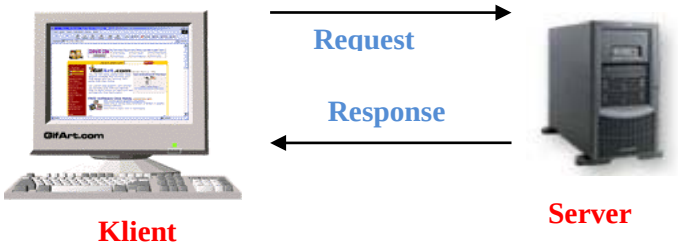
 nebo <img />

Odpověď na kontrolní otázku 1.6

Dokument vždy obsahuje párové elementy `<html> </html>`, `<head></head>` a `<body> </body>`. Html dokument obsahuje hlavičku a tělo. V těle dokumentu je hlavní informační obsah, v hlavičce dokumentu je její název zobrazen mezi elementy `<title>` a `</title>`. Před samotným začátkem dokumentu je ještě definice `<!DOCTYPE >`, která podle specifikací W3C konsorcia udává typ dokumentu.

Odpověď na kontrolní otázku 1.7

Komunikace počítače s internetovým prohlížečem na straně jedné a webovým serverem poskytujícím výslednou webovou stránku na straně druhé probíhá v modelu klient/server. Tento model komunikace pracuje způsobem žádost/odpověď (*request/response*). V praxi to znamená, že klientský počítač zašle serverovému požadavek na informace. Serverový počítač klientovi odpoví a informace mu poskytne. Klientský počítač pak získané informace zobrazí v prohlížeči.



Obr. 13.2 Komunikace typu klient/server

Vše je tedy založeno na principu žádost/odpověď, neexistuje žádné trvalé spojení, jedná se o dva samostatně běžící procesy na počítačích. Server nemá žádné tušení o tom, co se právě na klientském počítači děje.

Odpověď na kontrolní otázku 1.8

Zdrojový kód stránek můžeme psát i v obyčejném textovém editoru jako je *notepad*. Mezi volně šiřitelné editory patří velice oblíbený editor PSPad. Existuje desítky html editorů, které lze freewarově používat. Na opačném konci produktů stojí například profesionální nástroj Dreamweaver společnosti Macromedia – dnes již Adobe a Microsoftími produkty Visual Studio a Web Expression, který je nástupcem bývalého nástroje Front Page z balíku Office této společnosti.

Klíč k řešení – kapitola 2

Odpověď na kontrolní otázku 2.1

Atributy jsou upřesňující formátující vlastnosti html elementů. Příslušný element v definici tagu v ostrých závorkách ihned po jeho názvu definujeme syntaxí vlastnost=“hodnota“. Atributů můžeme použít pro jeden element více, nezáleží přitom na jejich pořadí a oddělíme je pouze mezerou:

```
<název elementu atribut=“hodnota“ atribut=“hodnota“ >
```

Odpověď na kontrolní otázku 2.2

Pro zobrazení klíčových slov, popisu stručného obsahu stránek a informace o autorovi je určen tag *meta*. Můžeme použít následující kód:

```
<meta name=“keywords“ content=“klíčová slova “>
```

```
<meta name="author" content="jméno autora ">
```

Odpověď na kontrolní otázku 2.3

Barvy jsou vyjádřeny ve formátu RGB (*RedGreenBlue*) pomocí hexadecimálního kódu nebo přímo čísel odstínu od 0 do 255. Při použití hexadecimálního kódu použijeme před číselným vyjádřením barvy znak **#**. Definovat barvu tedy můžeme takto:

```
<body bgcolor="#C0C0C0" text="rgb(0,0,255) ">
```

Odpověď na kontrolní otázku 2.4

Podle typu příslušného elementu zvolíme atribut, kterým se nastavuje barva. Potom máme tři možnosti, neboť žlutá barva patří mezi základní barvy pojmenované jménem, proto můžeme použít *atribut_pro_nastavení_barvy*="yellow" nebo "rgb(255,255,0)" nebo "#ffff00".

Odpověď na kontrolní otázku 2.5

Použijeme atribut `target` elementu `<a>`. Do nového okna jej otevřeme takto:

```
target="_blank"
```

Odpověď na kontrolní otázku 2.6

Text elementu `<u>` prohlížeč zobrazí podtrženě (*underline*) a to bude pro uživatele matoucí, neboť v hypertextových dokumentech bývají podtržené odkazy. Text mezi elementy `<u> a </u>` bude podtržen, avšak samotným odkazem nebude.

Odpověď na kontrolní otázku 2.7

Některé starší elementy se nedoporučují ze tří důvodů. Prvním důvodem je zastaralost uvedeného elementu. Takový element nemusí být podporován prohlížeči a může dělat při zobrazení problémy. Takovým elementem je například `<font>`. Druhým důvodem pro nedoporučení používat elementy je pozbytí jejich významu při vývoji hypertextových dokumentů. Element `<u>` prohlížeč zobrazí podtrženě, avšak v hypertextových dokumentech bývají podtržené odkazy, ostatní podtržený text je matoucí. Třetím důvodem je nahrazení novějším elementem či dokonce CSS stylem.

Odpověď na kontrolní otázku 2.8

Použijeme atribut `valign` elementu `<td>`. Poté volíme vhodnou hodnotu atributu:

```
valign="middle"
```

Odpověď na kontrolní otázku 2.9

Použijeme atribut `rowspan` nebo `colspan` elementu `<td>`. Záleží, jestli chceme slučovat buňky vedle sebe nebo pod sebou. Jako hodnotu uvádíme počet spojených buněk:

```
rowspan="2"
```

Odpověď na kontrolní otázku 2.10

Tam, kde se chceme pomocí linku na stejné stránce odkázat, musíme umístit odkazy pomocí elementu `<a>` s atributem `name`. Odkazy na příslušné místo pak již budou pomocí atributu `href`, v uvozovkách však před jménem odkazu umístíme znak `#`.

```
<a name="zalozka"> text </a>
<a href="#zalozka"> odkaz na záložku </a>
```

Klíč k řešení – kapitola 3

Odpověď na kontrolní otázku 3.1

Pro použití CSS stylu známe tyto tři hlavní důvody:

- Použití CSS stylů ušetří čas.
- Stránky, které používají CSS styly mají jednotný vzhled,
- Vzniká nám přehledný a dobře citovatelný zdrojový kód.

První důvod oceníme jak při tvorbě jedné či několika stránek, tak i při tvorbě složitějšího webového portálu. Jistě se atributy HTML elementů opakují a samozřejmě z druhého uvedeného důvodu jsou často totožné. Proto je výhodnější vlastnosti elementů nadefinovat jednou hromadně a pak se na toto formátování odkazovat. Dokážeme si jistě představit tu dlouhou práci při editaci stránek, pokud bychom chtěli drobně upravit jejich vzhled. Editace atributů stovek elementů by mohla zabrat tolik času jako samotný návrh nového webu. Při změně barev, formátování, pozicování, velikosti a dalších atributů nám tedy stačí editovat pouze styl, který se aplikuje na jednotlivé elementy ve všech stránkách aplikace.

Odpověď na kontrolní otázku 3.2

Existují tři způsoby zápisu. Prvním z nich je tzv. *Inline zápis.* V postatě CSS style je atributem *style* Html elementu. Tímto způsobem si nijak nepomůžeme v šetření času, místem a přehledností, avšak jsou důvody, proč tento způsob používat. Jedním z nich je fakt, že *inline* zápis stylu dostane přednost před stylem v hlavičce. Nechceme-li tedy příslušný element formátovat podle hlavního stylu, můžeme to takto jednoduše vyřešit. Druhý způsob zápisu je tzv. *Interní zápis*. Tento zápis stylu se definuje v elementu `<head>`. Všechny použité elementy v html dokumentu pak získají uvedené vlastnosti definovaného stylu.
Nejčastěji se však volí *Externí zápis* kaskádového stylu. Obvykle je externí soubor s kaskádovým stylem ve stejném adresáři, či v nějakém podadresáři zdrojové aplikace a samotný odkaz na styl je proveden v každé zdrojové stránce v hlavičce za pomocí elementu `<link>`.

Odpověď na kontrolní otázku 3.3

Serif znamená patkové pásmo, Sans-serif bezpatkové. Typickým představitelem bezpatkového písma je Arial, zatímco patkovým písmem je Times New Roman.

Obr. 13.3 Rozdíl mezi bezpatkovým (vlevo) a patkovým písmem

Odpověď na kontrolní otázku 3.4

Selektor class aplikuje formát CSS stylu na více elementů a naopak pomocí tečkové konvence dokáže u jedoho elementu nadefinovat více typů css formátování. Použití je potom jednoduché Selektor třídy se použije jako atribut *class* html elementu.

Odpověď na kontrolní otázku 3.5

Pokud budeme chtít pomocí CSS stylu definovat vlastnosti elementu se kterým jsme se v CSS stylech dosud nesetkali, pomocí specifikace CSS stylu nebo výukových online materiálů, vyhledám příslušné formátování požadované vlastnosti html elementu.

Odpověď na kontrolní otázku 3.6

Prvním typem selektoru je selektor *id*. Při používání kaskádových stylů vznikla potřeba definovat nějaký styl s více vlastnostmi, který budeme aplikovat na určitý typ elementu. Tento styl je vhodné pojmenovat, neboli přiřadit jednoznačný identifikátor tzv. *id*, na které se pak budeme odkazovat. Id tak bude atributem html elementu.

Druhým selektorem je selektor třídy, který na rozdíl od selektoru *id* aplikuje formát CSS stylu na více elementů. Selektor třídy se používá jako atribut *class* html elementu a před definicí stylu jeho název předchází tečka. Oba selektory *id* i *class* ohraničují své vlastnosti a hodnoty ve složených závorkách.

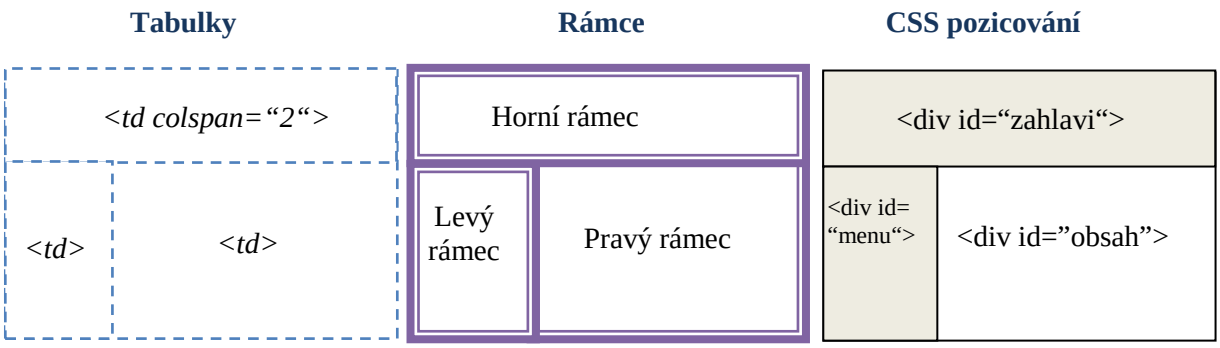
Odpověď na kontrolní otázku 3.7

Obecně platí, že CSS styly mají daleko širší možnosti použití než atributy HTML elementů. Atributy HTML elementů se zapisují jinak než vlastnosti CSS stylů, přesto, že vyjadřují mnohdy stejnou funkčnost viz(*align* vs. text-*align*).

Klíč k řešení – kapitola 4

Odpověď na kontrolní otázku 4.1

Rozložení obsahu stránek je řešeno různými způsoby, v různých dobách byly preferovány všechny. Prvním způsobem jak můžeme vytvořit rozložení obsahu na stránku do různých částí obrazovky je použití tabulky. Pomocí spojování buněk a nastavení velikosti řádku se pohodlně určil rozměr požadovaných míst pro rozložení na stránce. Tohle řešení mělo ale jednu nevýhodu. Tabulka se zobrazí, až se celá načte do prohlížeče. To znamená, že uživatel musí čekat, až se načtou všechny elementy včetně obrázků a to při pomalejším internetovém připojení může zabrat nějaký čas, který by při postupném načtení stránky již využil k jejímu čtení.



Obr. 13.4 Tři způsoby návrhu layoutu

Druhou možností bylo v roce 1996 zavedení rámců tzv. `<frame>`. Hlavní stránka neobsahovala element `<body>` ale definovala množinu rámů `<frameset>`. Množina rámů obsahuje již jednotlivé rámy `<frame>`, které ve svých atributech mají definovanou výšku a šířku. Ve výsledku pak máme navigační obsah v jednom rámu a veškerý informační obsah je směrován atributem *target* do hlavního rámu. Třetí možností, která se dříve nepoužívala, protože dvě předchozí byly jednodušší je pozicování pomocí CSS. V dnešní době již však většina moderních webů a portálů používá tento způsob. Je to

dáno nejen rozvojem specifikace CSS ale také neustále se vyvíjejícími komfortními nástroji pro práci s těmito technikami.

Odpověď na kontrolní otázku 4.2

V současné době je doporučováno navrhovat rozmístění objektů na stránce pomocí CSS pozicování. Tabulkový layout či dokonce rámy jsou zastaralými technikami. Prohlížeče je sice podporují, ale přesto způsobují menší či větší problémy při návrhu či zobrazení takto zpracovaných stránek.

Odpověď na kontrolní otázku 4.3

Pozicování elementů html nám umožňuje umístit objekt na stránku doslova, kam chceme. Pozicovat můžeme absolutně nebo relativně. Z názvu termínu *absolutně* vyplývá, že daný objekt se bude nacházet přesně v souřadnicích, které mu určíme. *Relativně* naopak znamená posunutí v různých směrech oproti běžné poloze.

Odpověď na kontrolní otázku 4.4

V tabulce 13.1 jsou definovány CSS vlastnosti pro pozicování elementů. Všimněme si, že vlastnost *left* určuje posunutí doprava, můžeme si tuto vlastnost vysvětlit jako posunutí v pixelech z levého okraje stránky.

Tab. 13.1 CSS definice pozicování

Vlastnost	Popis	Hodnoty
position	umístění elementu	absolute, fixed, relative, static, inherit
left	posunutí doprava	auto, *délka* v px, *délka* v %, inherit
top	posunutí dolů	auto, *délka* v px, *délka* v %, inherit
right	pozicování od pravého okraje	auto, *délka* v px, *délka* v %, inherit
bottom	pozicování od spodního okraje	auto, *délka* v px, *délka* v %, inherit
clip	oříznutí absolutně pozicovaného elementu	shape, auto, inherit
overflow	přetékání nebo oříznutí elementu	auto, hiden, scroll, visible, inherit
z-index	definice překrývání elementu (v ose z)	auto, *číslo*
visibility	viditelnost elementu	visible, hidden

Odpověď na kontrolní otázku 4.5

Element `<div>` znamená definici určité části definované jako celý oddíl, mnohdy je v tomto elementu celý informační obsah stránky. Nejčastěji element používáme v souvislosti s definicí kaskádového pozicování. Výsledná úprava spočívá v nastavení elementů `<div>` se selektorem id nadefinovaným v kaskádovém stylu.

Odpověď na kontrolní otázku 4.6

1. Vyhledáme vhodné menu pro naše webové stránky, to můžeme udělat ve vyhledávači volbou vhodných klíčových slov, např. *free css menu*.
2. Zvolíme volbu vertikální nebo horizontální menu, a vybereme pro naši potřebu vhodné menu.

3. Zvolené menu může obsahovat obrázky, proužky či jinou grafiku, kterou je nutné uložit do našeho projektu.
4. Zvolené menu obsahuje zdrojový kód CSS stylu, který importujeme do externího souboru, nebo přímo do zdrojového kódu naší stránky.
5. Pro rychlé otestování menu je nutné použít i zdrojový kód html. Pokud jej nepoužijeme, musíme definovat vlastní elementy `<div>` se selektorem id nadefinovaným v kaskádovém stylu.
6. Zdrojový kód menu upravíme tak, aby vyhovoval našim potřebám, tzn., upravíme názvy odkazů, jejich počet a cílové URL odkazů.

Klíč k řešení – kapitola 5

Odpověď na kontrolní otázku 5.1

Intellisense je vestavěný nástroj ve vývojových prostředích Microsoft jako Visual Studio .NET či Microsoft Expression Web, který umožňuje přímo při psaní zdrojového kódu využívat další předvyplněný zdrojový kód, možné vlastnosti elementů a CSS, které během psaní kódu postupně zpřesňuje a doplňuje nabízené možnosti dle standardů. Nástroj Intellisense ukončuje párové elementy. Vývojář tak má jistotu, že nepoužije nevhodný či nestandardizovaný zdrojový kód, jeho práce je navíc snadnější a rychlejší.

Odpověď na kontrolní otázku 5.2

W3C konsorcium nabízí online nástroje pro validaci vašich vytvořených web stránek. Jsou dostupné z více míst, například CSS validátor je dostupný z URL adresy http://jigsaw.w3.org/css-validator/, xhtml validátory z adresy http://validator.w3.org/.
Pomocí těchto služeb si můžeme ověřit, zda-li naše stránky odpovídají standardům CSS a XHTML.

Odpověď na kontrolní otázku 5.3

Validaci xhtml nebo css můžeme provést uploadováním stránky se zdrojovým kódem, vložením zdrojového kódu přímo do formuláře nebo zadáním URI adresy, pokud už je stránka na webovém serveru. V případě úspěšné validace uvidíme výstup se zelenou lištou a oznámením, že žádné chyby nebyly nalezeny, v opačném případě bude lišta červená s hlášením o počtu chyb. V tomto případě uvidíme výpis jednotlivých chyb a varování s číslem řádku zdrojového kódu. Poté uvedené chyby vyhledáme a pokusíme se je dle standardů a specifikací opravit. Uvedený postup opakujeme.

Odpověď na kontrolní otázku 5.4

Pokročilá vývojová prostředí dokáže současně pod sebou zobrazit pracovní plochu návrhu stránek a jejich zdrojový kód. To je zajištěno pomocí přepínačů *Split – Design –Code* na hodnotu *Split*. Tento přepínač je většinou umístěn na pracovní ploše pod oknem návrhu ve formě záložkového menu.

Klíč k řešení – kapitola 6

Odpověď na kontrolní otázku 6.1

- Dokáže reagovat na události jako klikání myší, změna polohy objektů apod.
- Pomocí něj můžeme validovat data z formulářů na straně klienta.

- Dokáže vytvořit dynamický text v HTML stránce.
- Dokáže přizpůsobovat a měnit HTML elementy.
- Umožňuje číst a zapisovat proměnné cookies, detekovat klientův prohlížeč a další užitečné vlastnosti.

Odpověď na kontrolní otázku 6.2

Javascript pracuje na straně klienta, nelze tedy řídit žádné události na serveru, javascript funguje pouze v prohlížeči, uživatel jej může zakázat. Rovněž přístup k systémovým objektům a souborům mimo cookies je omezen.

Odpověď na kontrolní otázku 6.3

Modulo znamená zbytek po dělení. Operátorem operace modulo je znak %.

Odpověď na kontrolní otázku 6.4

5 % 3 = 2
6 % 2 = 0

Odpověď na kontrolní otázku 6.5

Událost je situace, která se stane na klientské straně, například stránka se načte do prohlížeče, uživatel klikne myší, změní se vlastnost nějakého prvku. Na tyto situace můžeme reagovat pomocí skriptu. Nejčastějšími událostmi jsou onLoad, onClick, onChange ale také události klávesnice, události formuláře, události myši např. onMouseOver, onMouseMove , onMouseDown, onMouseUp, onMouseMove, onMouseOut a další.

Odpověď na kontrolní otázku 6.6

Javasript je částečně objektový jazyk, nevyužívá možnosti objektově orientovaného programování jako pokročilé OOP jazyky. U javascriptu se pod pojmem objektový chápe jisté uspořádání systému a přístup k objektům pomocí metod a vlastností. K objektům a podobjektům se přistupuje tečkovou konvencí rovněž metoda objektu je volána způsobem objekt.metoda(). Pokud budeme mít v dokumentu formulář obsahující textbox s hodnotou proměnné, pak se podle tečkové konvence na tuto proměnnou budu odkazovat:

dokument.formulář.textbox.proměnná.její_hodnota

Odpověď na kontrolní otázku 6.7

Nejčastějšími objekty, se kterými javascript pracuje je object *document*, *Date* a *Math*. Objekt Date nabízí metody pro zjištění data a času avšak vrací pouze číslo dne v týdnu, měsíce v roce, apod., nikoli textovou podobu data. Rovněž u času vrací počet hodin, minut a sekund od počátku jednotky, nikoliv celý čas. Objekt Math využíváme pro operace, na které nám nestačí běžné aritmetické operátory, avšak takových výpočetních operací na webové stránce není příliš zapotřebí. Spíše využijeme metod zaokrouhlování či druhou odmocninu.

Odpověď na kontrolní otázku 6.8

Metoda getMonth() objektu Date vrací měsíc v roce, ale pozor je to číslováno od nuly tudíž Leden = 0, prosinec = 11. Ale protože pole číslujeme rovněž od nuly, tak je tento fakt spíše výhodou.

Klíč k řešení – kapitola 7

Odpověď na kontrolní otázku 7.1

Elementy komunikace v počítačové síti jsou zařízení, které mezi sebou komunikují, zatímco komponenty sítě zahrnují také systémový software a služby, které běží na těchto síťových zařízeních.

Odpověď na kontrolní otázku 7.2

Mezi koncová zařízení patří:

- PC, Notebook, PDA
- Server, (DB server, Web server, file server)
- Síťová tiskárna
- VoIP telefon
- Kamery
- Další koncová zařízení (čtečky kódu, scannery, atd.).

Odpověď na kontrolní otázku 7.3

Síťová zařízení můžeme nazvat zprostředkující (ang. *intermediary*), neboť zprostředkovávají komunikaci koncovým zařízením v síti.

Odpověď na kontrolní otázku 7.4

V dnešní době se stále nejvíce setkáváme s komunikačním médiem zvaným kroucená dvoulinka, neboť v administrativních a veřejných prostorách je pro své vlastnosti nejvhodnější. Můžeme se také setkat s koaxiálním kabelem nebo dokonce s optickým vláknem, které je ale pro svou vysokou cenu implementován spíše na vysokorychlostní páteřní síť. S rostoucí popularitou bezdrátového připojení nelze opominout ani tento typ přenosu, který využívá jako komunikační médium vzduch.

Odpověď na kontrolní otázku 7.5

V metalickém provedení přenosového média je signál generován elektrickými impulsy podle nějakého předepsaného kódování. Při optickém přenosu jsou data přenášena světelnými pulsy. Bezdrátový přenos je realizován pomocí elektromagnetických vln.

Odpověď na kontrolní otázku 7.6

Propojení mezi jednotlivými lokálními sítěmi zajišťuje nějaký externí subjekt, kterému se říká *telekomunikační provider*. Telekomunikační provider je zodpovědný za provoz rozsáhlé sítě od určitých bodů propojení lokálních sítí.

Odpověď na kontrolní otázku 7.7

Počítačové sítě můžeme rozdělovat dle mnohých kritérií. Od jednoduchých kritérií daných počtem uživatelů, rychlostí sítě, typu přenosového média a dalších faktorů až po škálovatelná kritéria, která dělí sítě na veřejné, privátní, pronajaté atd. Nejčastěji rozdělujeme sítě na lokální a rozsáhlé. Lokální počítačová síť pokrývá jedno souvislé území a je plně pod správou jedné organizace včetně poskytovaných služeb a zajištění bezpečnosti sítě. Rozsáhlá počítačová síť propojuje sítě lokální a od těchto bodů připojení je pod plnou správou telekomunikačního providera.

Odpověď na kontrolní otázku 7.8

Komunikační protokol je nástroj, který je zodpovědný za správné doručení zprávy ze zdrojového do cílového zařízení. Komunikační protokoly mají na starost:

- Zajistit vhodný formát a strukturu zprávy
- Vytvořit cestu přes správná síťová zařízení
- Zajistit počátek a konec konverzace
- Detekovat možné příčiny chyb a informovat o tom příslušná zařízení

Odpověď na kontrolní otázku 7.9

Dvě zařízení komunikují v počítačové síti prostřednictvím paketů. Formát paketu se po síťové cestě bude měnit, právě podle toho jaký protokol ho bude mít na starost. Obecný tvar však vždy vypadá takto, paket musí obsahovat adresu zdrojového počítače, cílového počítače a vlastní data pro přenos. Každá konverzace mezi dvěma koncovými zařízeními s sebou nese stovky či tisíce paketů, které mají shodnou počáteční a cílovou adresu avšak jiná data a úkolem protokolů je tyto data dovést správně do cíle.

Klíč k řešení – kapitola 8

Odpověď na kontrolní otázku 8.1

Vzájemnou interakcí protokolů se vytváří tzv. protokolový zásobník (*Protocol stack*). Ten v sobě obsahuje jednotlivé vrstvy protokolů, hierarchicky právě od aplikačních dat až po zmíněné nuly a jedničky na nejnižší vrstvě. Například při komunikaci PC s webovým serverem se jako aplikační protokol konverzace účastní protokol *HTTP*, protokolem vrstvy transportní je *TCP*. V jiných případech komunikace po počítačové síti se interakce protokolů účastní jiné protokoly, např. při odesílání e-mailové zprávy je aplikačním protokolem *SMTP*, naopak při jejím přijímání to může být *POP3*, při prohlížení např. streamovaného videa je transportním protokolem *UDP*.

Odpověď na kontrolní otázku 8.2

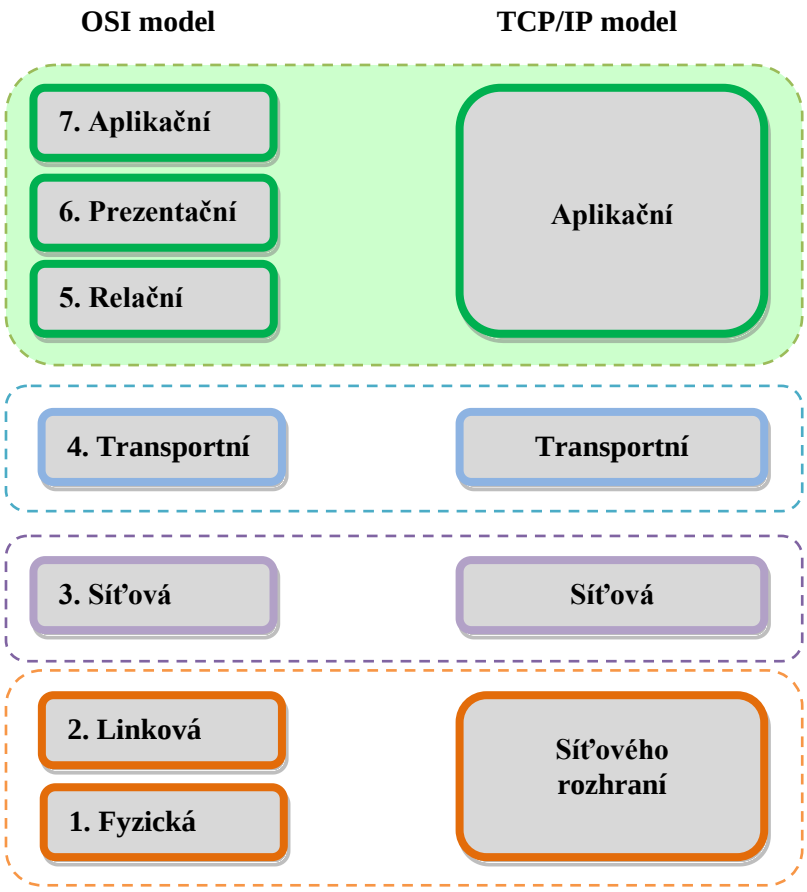
Protocol Stack je protokolový zásobník, který v sobě obsahuje 4 vrstvy TCP/IP modelu: aplikační, transportní, síťovou a vrstvu síťového rozhraní. Tyto 4 vrstvy zabalují a rozbalují na své cestě datový paket.

Odpověď na kontrolní otázku 8.3

Tento model zavádíme proto, že pomocí něj definujeme úkoly každé vrstvy. To umožňuje výrobcům odlišných technologií vytvářet protokoly podle standardů tak, aby protokoly na jednotlivých vrstvách si dokázaly porozumět. Další výhodou vrstevnatého modelu je fakt, že změna na jedné vrstvě neovlivní vrstvy ostatní, ani nad, ani pod vrstvou, kde ke změně došlo.

Odpověď na kontrolní otázku 8.4

Čtyřvrstvý TCP/IP model odpovídá sedmivrstvému referenčnímu modelu tak, že aplikační vrstva slučuje funkce aplikační, prezentační a relační referenčního modelu a vrstva síťového rozhraní slučuje fyzickou a linkovou vrstvu. Transportní a síťová vrstva si v obou modelech odpovídá. Funkčnost sloučených vrstev tak přebírá pro horní vrstvy aplikační vrstva a pro spodní vrstvy vrstva síťového rozhranní.



Obr. 13.5 Srovnání obou síťových modelů

Odpověď na kontrolní otázku 8.5

1. Vytvoří se uživatelská data na zdrojovém zařízení, například vyplněním webového formuláře nebo napsáním elektronické pošty.
2. Následuje tzv. proces segmentace, kdy se jednotlivá data rozdělí na informační celky, zároveň se provede zapouzdření neboli enkapsulace do paketů, které se na jednotlivých vrstvách TCP/IP modelu předávají nižším vrstvám.
3. Následuje generování impulsů na přenosovém médiu, které reprezentuje příslušná data.
4. Dále se transportují data po síti přes mnoho síťových zařízení s příslušným komunikačním médiem.
5. Nakonec data dorazí k cílovému zařízení, kde jsou přijata spodní vrstvou síťového rozhraní.
6. Následuje proces dekapsulace, neboli rozbalení paketu na jednotlivých vrstvách TCP/IP modelu.
7. Cílová data jsou předány aplikační vrstvě a konkrétní aplikaci, pro kterou jsou určena.

Odpověď na kontrolní otázku 8.6

Enkapsulace resp. dekapsulace je proces zabalení resp. rozbalení jednotky PDU na jednotlivých vrstvách sítě. Každý protokol zapouzdří data přijatá z vyšší vrstvy a opatří svůj PDU o hlavičku. Při dekapsulaci je proces opačný. Nejprve je oddělena hlavička s příslušnými údaji a data jsou poskytnuta vyšší vrstvě.

Odpověď na kontrolní otázku 8.7

Formát protokolové jednotky na jednotlivých vrstvách je následující:

- Data – Na aplikační vrstvě je pro PDU používán termín data.
- Segment – Transportní vrstva přidá na začátek hlavičku svého konkrétního protokolu.

- Paket – na síťové vrstvě je k segmentu z předchozí vrstvy přidána na začátek hlavička síťového protokolu.
- Rámec – Na úrovni vrstvy síťového rozhraní protokol přibaluje nejen hlavičku ale také zakončení rámce.
- Bity – Když je rámec spodní vrstvou překódován na posloupnost bitů, je vše připraveno na vyslání po komunikačním médiu.

Odpověď na kontrolní otázku 8.8

Pojmy ***Header*** – hlavička a ***Trailer*** - zakončení jsou anglické názvy pro části PDU. Zatímco svou vlastní hlavičku zpracovává každá forma PDU: rámec paket i segment, zakončení má pouze rámec na spodní vrstvě síťového rozhraní.

Odpověď na kontrolní otázku 8.9

Jednotlivým informačním úsekům, pro které je běžné používat obecný termín paket, bychom měli říkat protokolová datová jednotka (*Packet Data Unit*). Tento termín je zaveden z důvodu zapouzdřovacího procesu. Tato datová jednotka (*PDU*) se na každé vrstvě jmenuje jinak a proto by bylo nesprávné používat termín paket, neboť to je již tvar PDU na síťové vrstvě.

Klíč k řešení – kapitola 9

Odpověď na kontrolní otázku 9.1

Aplikační vrstva je horní vrstvou obou síťových modelů, její úlohou je zajistit rozhraní mezi aplikacemi spuštěnými na jednotlivých zařízeních, které potřebují komunikovat po počítačové síti a nižšími vrstvami, které zprostředkovávají síťové a přenosové funkce.

Odpověď na kontrolní otázku 9.2

Aplikační vrstva modelu TCP/IP v sobě zahrnuje funkcionalitu všech tří horních vrstev OSI modelu. Prezentační vrstva OSI modelu definuje úlohu vlastního kódování a konverze aplikačních dat. Úkolem relační vrstvy je implementovat funkce udržování dialogu mezi zdrojovou a cílovou aplikací. Podstatou je udržovat dialog, v případě chyb jej obnovit či ukončit.

Odpověď na kontrolní otázku 9.3

Hlavním úkolem protokolu aplikační vrstvy je výměna dat mezi aplikacemi běžícími na zdrojovém a cílovém zařízení komunikace. Aplikační protokoly definují procesy na obou koncích komunikačního řetězce, typy vyslaných zpráv a jejich syntaxi, způsob odesílání dat a podobu očekávané odpovědi a také způsob spolupráce s nižší vrstvou síťového modelu.

Odpověď na kontrolní otázku 9.4

Aplikační protokoly pracují na komunikačním modelu klient/server. V modelu klient/server se zařízení, které se dožaduje informace, nazývá klient a zařízení, které na požadavky odpovídá, se nazývá server. Tento model komunikace je navazován právě v aplikační vrstvě. Klient zahajuje komunikaci vysláním požadavku, na který server odpovídá.

Odpověď na kontrolní otázku 9.5

Proces přidělení IP adresy se řídí komunikace mezi PC v roli klienta a DHCP serveru. Jsou celkem zapotřebí čtyři komunikační fáze, než počítač obdrží IP adresu. Po připojení počítače do sítě počítač

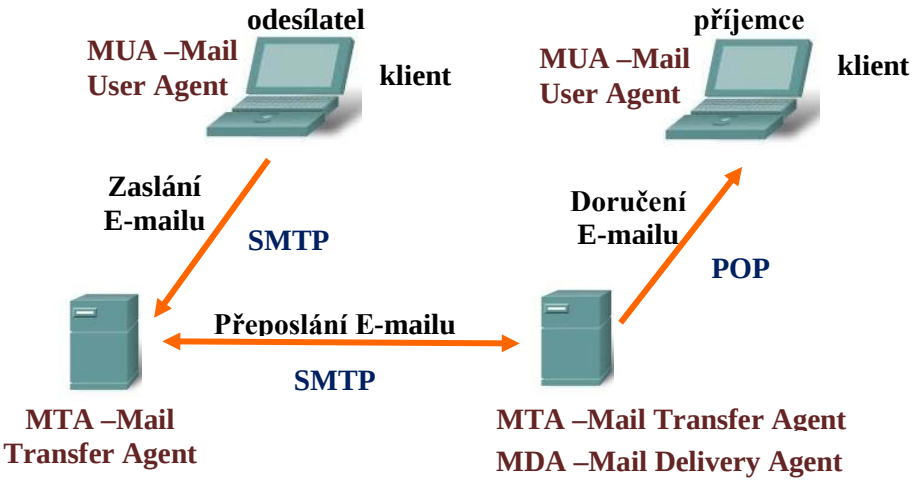
vyšle *broadcastový* paket tzv. DHP Discover, který hledá přítomnost DHCP serveru. *Broadcastový* znamená fakt, že je vyslán všem účastníkům sítě. Počítač totiž neví komu má žádost poslat, tak nejjednodušší je poslat paket všem. Tento způsob rozeslání žádosti není v počítačové síti neobvyklý, používá jej mnoho protokolů na různých vrstvách. *Broadcast* paket dojde jednotlivým zařízením a zařízení kterému je určen ho zpracuje, ostatní zařízení paket zahodí. Discover Paket tedy dojde na DHCP Server a ten odpoví paketem DHCP Offer, který obsahuje nabídku IP adresy, masky sítě, nastavení DNS serveru a výchozí brány. Klient dalším požadavkem oznamuje, že s přidělenými údaji souhlasí a na závěr přijde ze serveru potvrzení. Tohle je samozřejmě ideální situace. Pokud se klient a server na přidělených údajích nedomluví, nebo přijde ze serveru zamítnutí (místo **ACK** přijde paket **NACK**), celý proces se musí opakovat.

Odpověď na kontrolní otázku 9.6

Webová adresa je v textovém formátu, avšak počítač potřebuje jednoznačný údaj o adrese webového serveru a tak potřebujeme prostředníka, který převede jmenný název na IP adresu. Tímto prostředníkem se stává DNS server, tzv. doménový jmenný systémový server. Celý systém názvu webových adres je založen na komunikaci doménových serverů, které mají odkazy na další názvy domén. Doménami prvního řádu jsou národní domény (např. .cz .sk), neomezené domény (.org, .com, .net, .info), speciální domény (.edu, .gov, .biz) a další. Pod těmito doménami se nacházejí domény druhé úrovně, atd. Jmenné servery jsou autoritativní, to znamená, že mají informace o názvech serverů v příslušných doménách. Poslední autoritativní jmenný server poskytne IP adresu pro přeložení ze jména uloženého v URL adrese.

Odpověď na kontrolní otázku 9.7

Elektronická pošta je jednou z nejoblíbenějších a nejčastěji používaných služeb počítačových sítí. Celý proces posílání pošty se opět řídí modelem klient/server s tím, že klient je v roli jak odesílatele pomocí protokolu SMTP (*Simple Mail Transfer protocol*), tak v roli příjemce, kdy poštu přijímá pomocí protokolu POP (*Post office Protocol*) nebo IMAP(*Internet Message Access Protocol*). Klientovi se říká ***Mail User Agent*** (*MUA*). Poštovní server se účastní dvou rolí, buďto má klienta ve svém seznamu příjemců a zprávu doručí, v tomto případě je jeho role ***Mail Delivery Agent*** (*MDA*) anebo zprávu přepošle dalšímu emailovému serveru. V tomto případě je jeho role ***Mail Transfer Agent*** (*MTA*).



Obr. 13.6 Proces zaslání a příjmu pošty pomocí MUA, MTA a MDA

Odpověď na kontrolní otázku 9.8

Protokol FTP se liší od ostatních tím, že potřebuje pro svou činnost dvě spojení. Jedno spojení určuje řízení komunikace a druhé spojení je pro samotná data. Proto si TCP spojení alokuje dva porty 20 a 21.

Odpověď na kontrolní otázku 9.9

Transportní vrstva zajišťuje segmentaci dat a jejich řízení nezbytně nutné pro různé typy komunikačních proudů. Její hlavní úlohy jsou: Řízení a sledování komunikace mezi aplikacemi na obou koncích, segmentace dat a řízení každého segmentu, složení aplikačních dat zpět z proudů segmentů a identifikace různých typů aplikací.

Odpověď na kontrolní otázku 9.10

Jejich úloha je totožná, je to řízení komunikace mezi aplikacemi. Rozdíl je však ve funkčnosti těchto protokolů. UDP je jednodušší protokol, který používá méně složitou strukturu PDU nazvanou datagramy. Datagramy jsou oproti segmentům TCP kratší, protože neobsahují sekvenční čísla a pole bytů pro potvrzení o přijetí paketu. Protokol UDP se používá v komunikaci mezi aplikacemi, které vyžadují rychlý přenos, který může být nespolehlivý. Proto není každý segment potvrzován. V praxi to znamená, že pokud konzumujete proud dat, např. Streamované video či hlasové služby po internet a nějaký datagram nedorazí, komunikace pokračuje dále, nemá vůbec žádný smysl se pokoušet o znovuzaslání nepřijatých dat. Naopak v aplikacích, které potřebují spolehlivost je využíván protokol TCP, který zajišťuje potvrzení příjmu segment a ztracená data posílá znovu.

Odpověď na kontrolní otázku 9.11

Porty dělíme do třech skupin. Tzv. dobře známé porty (well known ports) jsou čísla přidělovaná službám nebo aplikacím, které dobře známe jako např. poštovní služby či HTTP. Tyto porty jsou v rozsahu 1 -1023. Mezi dobře známé TCP porty patří: **20** FTP, **21** FTP, **23** Telnet, **25** SMTP, **53** DNS, **80** HTTP, **110** POP3, **443** HTTPS a další.

Odpověď na kontrolní otázku 9.12

Adresování pomocí portů je jedním z nejdůležitějších poslání transportní vrstvy. TCP a UDP protokoly zajišťují komunikační přenos segmentů a datagramů mezi různými aplikacemi. Právě čísla portů jednoznačně identifikují komunikující strany. Zdrojový port je asociován s aplikací na straně klienta neboli lokálního uživatele, cílové číslo portu je asociováno s aplikací na straně hosta nebo serveru na vzdálené straně. Porty jsou přiřazovány různými způsoby v závislosti na tom, zda-li se jedná o požadavek či odpověď. Zatímco server asociuje statické porty, klient dynamický přiděluje portům čísla pro každou konverzaci.

Klientský počítač vyslal na web server a poštovní server požadavek, inicializoval tedy komunikaci s dynamicky přiděleným portem. Např. klientský požadavek z IP adresy 158.196.140.10 bude opatřen zdrojovým portem 49152, resp 49956 u SMTP komunikace. Výsledná komunikace tak bude u http: 158.196.140.10:49152 → 172.16.1.2:80 a u smtp: 158.196.140.10:49556 → 172.16.1.9:25. Odpovědi pak budou zasílány zpět na odpovídající sokety - u http: 172.16.1.2:80 → 158.196.140.10:49152 a u smtp: 172.16.1.9:25 → 158.196.140.10:49556.

Klíč k řešení – kapitola 10

Odpověď na kontrolní otázku 10.1

IP adresa má jednoznačný 32 bitový formát (IP protokol verze 4). Adresu s tímto formátem v binární podobě před samotnou konverzí na dekadický formát rozdělíme na čtyři oktety. Každý oktet je oddělen tečkou a jeho hodnota se logicky může pohybovat v rozmezí 0 až 255.

Odpověď na kontrolní otázku 10.2

IP adresy jsou rozděleny do pěti tříd A-E viz tabulka:

Tab. 13.2 Třídy IP adres

Třída adres	Dekadický tvar prvního oktetu	Bitový rozsah prvního oktetu	Části IP adresy (N- síť, H –host)	Defaultní maska	Počet možných sítí a jejich hostů
A	**1-127**	**00000000 - 01111111**	**N.H.H.H**	**255.0.0.0**	**128 sítí 16777214 hostů/síť**
B	**128-191**	**100000000 - 10111111**	**N.N.H.H**	**255.255.0.0**	**16384 sítí 65534 hostů/síť**
C	**192-223**	**11000000 - 11011111**	**N.N.N.H**	**255.255.255.0**	**2097150 sítí 254 hostů/síť**
D	**224-239**	**11100000 - 11101111**	Multicast adresy		
E	**240-255**	**11110000 - 11111111**	Experimentální adresy		

Třídy adres jsou charakteristické hodnotou prvního oktetu (v tabulce červeně označené bity jsou konstantní). Třídy tímto způsobem určí hodnoty adresních rozsahů pro tři typy velikostí sítí s různým počtem připojených zařízení (viz. Poslední sloupec tabulky). Třídy adres D a E mají speciální použití.

Odpověď na kontrolní otázku 10.3

Maska sítě je opět 32bitové číslo ve tvaru IP adresy rozdělené do 4 oktetů. Můžeme se setkat i se zkráceným zápisem síťové masky, tzv. prefixem. Ten se určuje počtem jedničkových bitů v masce, takže prefix /24 odpovídá masce 255.255.255.0.

Odpověď na kontrolní otázku 10.4

Maska sítě má široký rozsah uplatnění, pomocí ní rozdělujeme sítě na podsítě, určujeme příslušnost ke konkrétní síti, počítáme adresy sítě atd. S maskou sítě pracují síťová zařízení, pomocí masky sítě se provádí směrování a mnoho dalších událostí.

Odpověď na kontrolní otázku 10.5

Mezi speciální typy adres můžeme zařadit multicastové adresy z adresového rozsahu 224.0.0.0 – 239.0.0.0, broadcastovou adresu 255.255.255.255 a opačnou adresu 0.0.0.0. Tato adresa se používá na routerech k definici defaultní cesty. Speciální adresou je také adresa 127.0.0.1. Touto adresou se označuje tzv. *loopback* neboli cesta zpět na sebe sama.

Dalšími adresy, které se dají vyčlenit pro své vlastní použití, jsou adresy privátní. Privátní adresy se vyskytují v těchto blocích (závorce vidíme zkrácený zápis rozsahu pomocí adresy sítě a její masky):

- 10.0.0.0 - 10.255.255.255 (10.0.0.0 /8)
- 172.16.0.0 - 172.31.255.255 (172.16.0.0 /12)
- 192.168.0.0 - 192.168.255.255 (192.168.0.0 /16)

Odpověď na kontrolní otázku 10.6

NAT znamená překlad adres – *Network Address Translation* a umožňuje celou privátní síť s mnoha počítači s privátní adresou zaštítit jednou veřejnou IP adresou nebo rozsahem veřejných adres. Pokud počítač z privátní sítě komunikuje ven ze své sítě, je mu překladem adres poskytnuta veřejná adresa.

Odpověď na kontrolní otázku 10.7

Při adresování v počítačových sítích se používají tři typy komunikace – unicast, broadcast a multicast: Unicast je proces vyslání paketu jednomu hostu, broadcast je proces vyslání paketu všem hostům v síti a multicast je proces vyslání paketu vybrané skupině hostů.

Odpověď na kontrolní otázku 10.8

Broadcast komunikace v principu pracuje tak, že daný paket je vyslán všem účastníkům sítě. Broadcast komunikaci nejčastěji využívají protokoly, které něco hledají, neznají adresu svého cíle, ale mají pro něj zprávu. Obsah paketu je tvořen informacemi, které cílový adresát pozná, a ostatní účastníci paket zahazují. S broadcast paketem se setkáváme například při DHCP komunikaci.

Klíč k řešení – kapitola 11

Odpověď na kontrolní otázku 11.1

Hlavním úkolem síťové vrstvy je zajištění výměny dat mezi jednotlivými uživateli sítě. Zajištění správného chodu této komunikace se skládá ze čtyř základních procesů: adresování, enkapsulace, směrování a dekapsulace.

Odpověď na kontrolní otázku 11.2

Síťová vrstva zajišťuje proces zabalení segmentu z nadřazené transportní vrstvy. K tomuto segmentu je přidána IP hlavička, proces enkapsulace tak vytvoří paket síťové vrstvy. IP hlavička obsahuje zdrojovou a cílovou IP adresu komunikačního procesu. Po ukončení enkapsulačního procesu je paket předán nižší vrstvě síťového modelu. Enkapsulace je tak prováděna na třetí vrstvě ISO/OSI modelu, proto je prováděna buďto routerem nebo L3 Switchem, tedy přepínačem, který umí pracovat s IP adresami na třetí vrstvě síťového modelu.

Odpověď na kontrolní otázku 11.3

IP protokol nemá mechanismus kontroly, zda-li paket dorazil do místa určení. Fakt, že adresát obdržel data, musí zajistit jiný protokol, v tomto případě protokol TCP transportní vrstvy.

Odpověď na kontrolní otázku 11.4

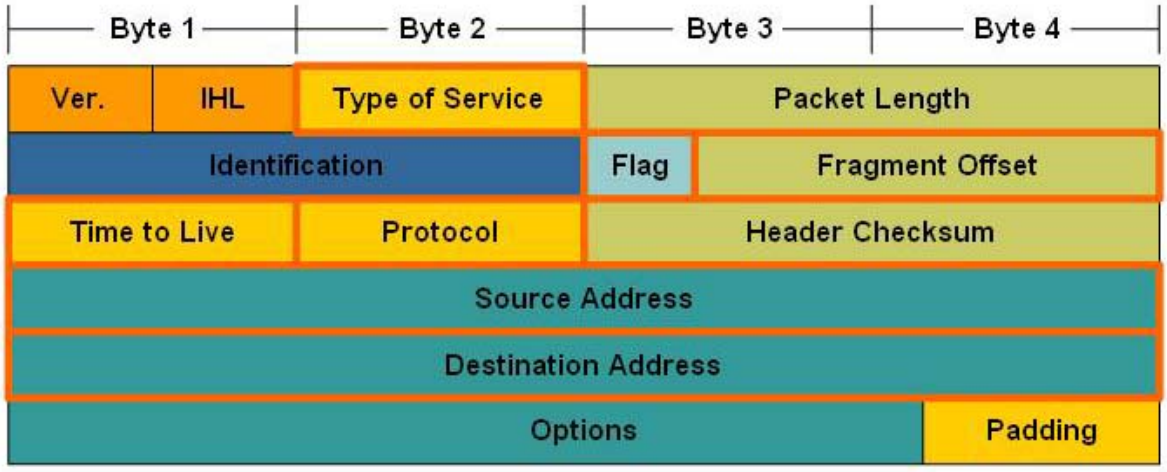
Základní charakteristikou IP protokolu jsou tři termíny: Nespojovaný (*Connectionless*), nezávislý na médiu (*Media Independent*) a s nejlepším úsílím (*Best Effort*). Tyto termíny nebo slovní spojení charakterizují podstatu protokolu. Nespojovaný v praxi znamená, že před samotným vysláním paketu není navazováno žádné spojení mezi zdrojovým a cílovým zařízením. Pojem „S nejlepším úsilím“, znamená fakt, že routery hledají nejlepší cestu pro každý vyslaný paket a nezávislost na médiu neřeší fyzickou vrstvu mezi komunikujícími zařízeními.

Odpověď na kontrolní otázku 11.5

Nejdůležitější položky IP hlavičky jsou zdrojová a cílová IP adresa. Další položky hlavičky paketu jsou:

- Verze: verze síťového protokolu IPv = 4.
- IHL: délka hlavičky v 32bitových slovech (4B) IHL= 5 znamená délku 20B
- Type of Service: osmibitové číslo udávající prioritu paketu
- Packet Length: délka celého paketu včetně hlavičky a dat

- Identification, Flag, Fragment Offset: informační bajty identifikující fragmentaci paketu. V některých případech je paket pro médium příliš dlouhý a jeho menší délku musí zajistit právě fragmentace.
- Time to Live: životnost paketu (osmibitové číslo), dekrementaci provádějí routery
- Protocol: identifikace protokolu vyšší vrstvy:01 ICMP, 06 TCP, 17 UDP.
- Header Checksum: kontrolní součet, pokud neodpovídá, paket je poškozen a bude zničen.



Obr. 13.7 Formát IP hlavičky protokolu

Odpověď na kontrolní otázku 11.6

Termín brána odpovídá opravdovému vstupu do venkovní sítě. V případě, že hostitelský počítač vyšle paket s IP adresou mimo rozsah své sítě, vstupuje do komunikace výchozí brána sítě, která je na směrovači. Tento směrovač rozhodne o síťové cestě tohoto paketu podle své směrovací tabulky.

Odpověď na kontrolní otázku 11.7

Výchozí brána se konfiguruje na hostitelském PC. Pokud tato brána není nastavena, počítač může komunikovat pouze uvnitř vlastní sítě. IP adresa výchozí brány musí vždy být ve stejné síti jako IP adresa počítače. Tomu musí samozřejmě odpovídat správná síťová maska.

Odpověď na kontrolní otázku 11.8

Směrovací tabulka je tabulka IP adres sítí, do kterých router může vyslat paket, protože zná síťovou cestu. Paket je vyslán korespondujícím rozhraním routeru na svou další cestu.

Odpověď na kontrolní otázku 11.9

Router se rozhoduje o tom, kam vyšle paket na základě směrovací tabulky. Pokud cílové síti neodpovídá žádná položka ve směrovací tabulce je paket zničen, pokud existuje cesta do sítě, je vyslán na rozhraní přes které je síť dostupná. Pokud je více cest do jedné sítě, router se rozhoduje na základě údaje o důvěryhodnosti, tzv. *Administrative distance*. Administrativní distance přímo připojené cesty je 0, statické cesty 1 další AD podle typu routovacího protokolu 5 – 200, neznámá cesta je 255. Čím menší je AD, tím má cesta větší prioritu před ostatními.

Odpověď na kontrolní otázku 11.10

Směrovací tabulka kromě IP adres dostupných sítí a korespondujících rozhraní směrovače také uvádí typ cesty, zda-li je přímo připojena na rozhraní routeru, staticky vložená či naučena dynamickým routovacím protokolem.

Odpověď na kontrolní otázku 11.11

Statická cesta je konfigurovaná manuálně přímo na routeru. Proces vložení statické cesty je realizován příkazem*: ip route adresa maska rozhraní*, kde adresa a maska určuje vzdálenou síť, kterou staticky připojujeme a rozhraní určuje lokální rozhraní routeru, přes které se do sítě dostaneme.

Odpověď na kontrolní otázku 11.12

Výhodou statické cesty je vysoká priorita při výběru cesty při směrování při shodě více cest k cíli, avšak nevýhodou je fakt, že při změně topologie sítě zůstane ve směrovací tabulce špatný záznam, dokud ji administrátor sítě neupraví, router se bude stále řídit podle staré topologie sítě.Pokud se změní cesta do sítě pomocí dynamického routovacího protokolu, je cesta při změně routovacích tabulek pomocí tohoto protokolu aktualizována. Může být smazána, mohou být změněny její parametry jako metrika, administrativní distance, tudíž může být změněna priorita této cesty.

Klíč k řešení – kapitola 12

Odpověď na kontrolní otázku 12.1

Linková vrstva posílá po komunikačním médiu rámce, zajišťuje jejich adresování, označuje jejich začátek a konec a také detekuje kolize a chyby. Linková vrstva na komunikačním médiu řídí komunikaci mezi jednotlivými uzly, obsahuje kontrolní informace o tom, kdo s kým komunikuje, kdy komunikace začíná a končí, který uzel bude komunikovat poté a jaké chyby se mohou vyskytovat při komunikaci.

Odpověď na kontrolní otázku 12.2

Mezi nejznámnější protokoly linkové vrstvy, patří: Ethernet protokol, PPP – Point to Point Protocol, HDLC – High Level Data Link Control protokol, Frame Relay protocol, a ATM – Asynchronous Transfer Mode protocol. Některé protokoly jsou přizpůsobeny technologii LAN s mnoha uzly na kratší vzdálenost, jiné protokoly pro sítě WAN s méně uzly. Typickým protokolem WAN je PPP protokol, jež má pouze dva uzly- vysílač a přijímač, a proto není nutné, aby rámec obsahoval zdrojovou a cílovou adresu. Naopak protokol 802.11 Wireless LAN obsahuje ve své struktuře několik adres, neboť potřebuje detekovat více bezdrátových uzlů.

Odpověď na kontrolní otázku 12.3

Fyzická vrstva ISO/OSI modelu zajišťuje transport rámce linkové vrstvy na úrovni bitů. Fyzická vrstva implementuje 4 nejdůležitější součásti: Fyzická média a k nim asociované konektory, reprezentaci posloupnosti bitů na konkrétním médiu, kódování dat a kontrolních informací a zajištění vysílání a přijímání na síťových zařízeních. Fyzická vrstva reprezentuje signál pomocí metod *encoding* a *signaling*. *Encoding* je převod určitého proudu datových bitů na nějaký předefinovaný kód podle bitové šablony, které rozumí vysílající i přijímající uzel. *Signaling* je pak fyzická reprezentace těchto nul a jedniček pomocí nějaké signální metody, tedy generováním elektrického signálu nebo optických pulzů.

Odpověď na kontrolní otázku 12.4

Preambule	Cíl	Zdroj	Typ	Data	FCS
8 bytů	6 bytů	6 bytů	2 byty	46 – 1500 bytů	4 byty

Obr. 13.6 Rámec protokolu Ethernet

- **Preambule** se používá pro synchronizaci rámců, obsahuje ukončovací značku při časování.
- **Cílová adresa** je 48 bitová MAC adresa příjemce.
- **Zdrojová adresa** je 48 bitová MAC adresa odesílatele.
- **Typ** určuje protokol vyšší vrstvy, který obdrží data po rozbalení rámce.
- **Data** jsou v tomto případě PDU vyšší vrstvy, obvykle IPv4 paket.
- **FCS (Frame Check Sequence)** je kontrolní součet rámce pro detekci chyb.

Odpověď na kontrolní otázku 12.5

Ethernet protokol dokáže v jednom rámci pojmout až 1,5 KB uživatelských dat.

Odpověď na kontrolní otázku 12.6

MAC adresa neboli fyzická adresa je jednoznačným identifikátorem síťového interface každého zařízení. Cílová MAC adresa je rozhodujícím faktorem, zda-li cílové zařízení rámec přijme nebo zahodí. Formát MAC adresy je 48 bitový rozdělen na dvě poloviny. První polovinu přiřazuje výrobcům standard IEEE, druhou polovinou rozlišují své síťové karty sami výrobci. MAC adresu ve vašem počítači zjistíte známým příkazem *ipconfig /all*.

Odpověď na kontrolní otázku 12.7

Pro propojování zařízení můžeme vytvořit pravidlo tak, že si rozdělíme zařízení na dvě skupiny. V první skupině bude hub a switch a ve druhé počítač a router. Pokud budeme propojovat zařízení, které patří do stejné skupiny, použijeme křížový kabel, pokud propojíme zařízení z první skupiny se zařízením z druhé skupiny, použijeme kabel přímý.

Odpověď na kontrolní otázku 12.8

Server budeme chápat jako zařízení ze stejné skupiny jako PC. S ethernetovým rozhraním routeru ho tudíž připojíme kříženým kabelem.

Rejstřík

A

adresní prostor · 107, 108, 109, 112, 113, 114, 115
adresování · 100, 112, 113, 114, 117, 125
ANSI · 130
aplikační data · 82, 85, 87
ATM · 77, 130, 134, 158
atribut · 18, 23, 29, 35, 142, 143, 144, 145
atributy · 15, 18, 21, 23, 24, 26, 28, 144, 145

B

beztřídní adresování · 108, 115
bity · 86, 88, 152
broadcast · 97, 110, 115, 153, 156
browser · 8

C

Caracan · 8
CSMA/CA · 129
CSMA/CD · 129
CSS · 4, 17, 18, 21, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 56, 137, 138, 143, 144, 145, 146, 147
CSS template · 48, 53, 55, 56, 61, 68
cykly · 60, 68

Č

číselné soustavy · 106

D

data · 86, 88, 151, 152
datagramy · 99, 154
decapsulation · 86
default gateway · 120
dekapsulace · 85, 86, 88, 90, 117, 118, 125, 151, 156
DHCP · 8, 93, 97, 103, 108, 110, 140, 152, 156
DNS · 8, 91, 93, 94, 95, 97, 102, 103, 139, 153
dobře známé porty · 100, 104, 154
dotaz – odpověď · 8
dynamické porty · 100

E

EIA/TIA · 130, 132, 133, 134
elementy · 11, 12, 14, 15, 17, 18, 23, 26, 27, 29, 30, 31, 35, 36, 38, 39, 41, 43, 49, 50, 56, 58, 68, 71, 141, 142, 143, 144, 145, 147, 148
Encoding · 130, 134, 158
enkapsulace · 85, 86, 88, 90, 117, 125, 126, 151, 156
Ethernet protokol · 130, 134, 158, 159
Externí zápis · 27, 35, 144

F

FCS · 129, 130, 158, 159
firewall · 73, 78
formát paketu · 78, 150
frame · 77, 86
Frame Relay protocol · 130, 134, 158
FTP · 91, 96, 102, 104, 153

G

Gecko · 8

H

HDLC · 130, 134, 158
header · 86, 90, 120, 152, 157
hexadecimální kód · 16, 23, 143
HTML · 4, 9, 10, 12, 15, 16, 17, 23, 24, 26, 28, 35, 58, 68, 87, 137, 138, 141, 144, 145, 148
http · 8
http komunikace · 8
http protokol · 8
HyperText Markup Language · 10
Hypertext transfer protokol · 8

Ch

Chrome · 8

I

IEEE · 130, 131, 132, 134, 159
Inline zápis · 27, 35, 144
intellisense · 50, 56
interakce protokolů · 81, 82, 90, 150
interní zápis · 27, 35, 144
IP · 73, 75, 81, 82, 83, 84, 85, 87, 88, 90, 91, 93, 94, 95, 97, 102, 103, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 123, 125, 126, 139, 140, 150, 151, 152, 153, 154, 155, 156, 157
IPv4 · 117, 118
IPv6 · 118
ISO · 3, 10, 12, 81, 82, 83, 84, 90, 92, 111, 130, 134, 141, 156, 158
ITU · 130

J

jádro prohlížeče · 8
javascript · 58, 59, 60, 68, 148

K

kaskádové styly · 10, 12, 26, 35, 57, 137, 138
klient/server · 7, 8
kompilátor · 10, 141
komponenty počítačové sítě · 72
komunikace · 7, 12, 14, 71, 72, 77, 78, 80, 81, 82, 83, 84, 85, 88, 89, 91, 92, 93, 94, 95, 96, 97, 98, 102, 103, 104, 105, 110, 111, 115, 116, 117, 120, 125, 142, 149, 150, 152, 153, 156, 157
komunikační médium · 128
komunikační proces · 85, 100
křížový kabel · 133

L

LAN · 74, 76, 77
layout · 32, 38, 39, 40, 41, 42, 43, 45, 47, 48, 49, 52, 53, 56, 146
link · 17, 22, 23, 24, 27, 35, 144
linková vrstva · 128, 134
LLC · 129
lokální počítačová síť · 76, 78, 149

M

MAC · 3, 84, 87, 106, 111, 128, 129, 130, 131, 132, 134, 135, 136, 158, 159
Mail Delivery Agent · 96, 153
Mail Transfer Agent · 96, 153
Mail User Agent · 96, 153
maska sítě · 105, 108, 109, 116
menu · 11, 21, 38, 40, 41, 43, 44, 45, 47, 48, 49, 51, 52, 56, 146, 147
meta · 17, 23, 142, 143
Microsoft Expression Web · 9, 50, 51, 54, 55, 56, 147
Microsoft Internet Explorer · 8
Mozilla Firefox · 8
multicast · 108, 110, 111, 115, 155

N

NRZ · 130
NRZI · 130

O

odkazy · 22, 143
oktet · 105, 108, 115, 154
Opera · 8

P

paket · 77, 78, 82, 85, 86, 87, 88, 97, 110, 111, 117, 118, 119, 120, 121, 122, 125, 150, 152, 153, 156, 157
PDU · 86, 87, 88, 90, 151, 152
POP3 · 82, 96, 150
pozicování · 26, 39, 40, 41, 42, 43, 45, 48, 49, 52, 53, 58, 60, 61, 62, 63, 64, 144, 145, 146
PPP · 130, 134, 158
prefix síťové masky · 109
prohlížeč · 8, 10, 12, 18, 58, 68, 81, 82, 93, 141, 143, 148
protocol stack · 82, 88, 150
protokol · 12, 71, 77, 78, 80, 81, 82, 83, 91, 92, 96, 102, 103, 105, 112, 115, 117, 118, 119, 122, 124, 125, 126, 150, 151, 152, 154, 156
protokolový zásobník · 81, 82, 88, 150
přímý kabel · 133
PSPad · 9

R

rámec · 3, 86, 88, 129, 130, 134, 152, 158
referenční · 82
registrované porty · 100
request/response · 7
RGB · 16, 23, 143
RJ-45 · 132, 133, 134
router · 73, 74, 117, 121, 122, 157

S

Sans-serif · 29, 35, 144
segment · 86, 88, 151
segmentace dat · 98
selektor třídy · 29, 35, 144, 145
selektory kaskádových stylů · 28
Serif · 29, 35, 144
Server · 8
SGML · 10, 12, 13, 141
signaling · 130, 134, 158
síťová vrstva · 84, 88, 117, 150
skriptovací jazyky · 10, 12, 58, 91
směrování · 108, 112, 115, 117, 119, 121, 122, 123, 125, 127, 139, 140, 155, 156, 158
SMTP · 8, 82, 91, 96, 102, 150, 153
Standard Generalized Markup Language · 10
statická cesta · 123, 125, 158
switch · 73, 74, 78, 120

T

T568A · 132, 133, 134
T568B · 132, 133, 134
TCP · 81, 82, 83, 84, 85, 87, 88, 90, 91, 97, 99, 102, 104, 112, 115, 118, 120, 125, 139, 140, 150, 151, 152, 153, 156, 157
TCP/IP · 83, 84, 85, 88, 91
telnet · 91, 97, 98, 102
trailer · 86, 90, 152
transportní · 84, 86, 87, 88, 150, 151
transportní vrstva · 98, 154
Trident · 8

U

UDP · 82, 91, 99, 104, 120, 150, 157
Unicast · 110, 115, 156
URI · 54, 147
URL · 21, 22, 23, 43, 48, 54, 56, 57, 81, 95, 147, 153

V

validace · 54, 55, 57, 147
větvení · 60, 61, 68
výchozí brána · 120, 125, 126, 157

W

WAN · 76, 77, 79
web server · 73, 95, 149
WebKit · 8
webový prohlížeč · 8
webový server · 8
well known ports · 100, 154

X

XHML · 50, 56
XML · 50, 56, 137, 139, 141
XSLT · 50, 56

Z

zapouzdření · 3, 83, 86, 128, 151
zařízení · 8, 71, 72, 73, 74, 75, 77, 78, 79, 80, 82, 84, 85, 87, 88, 91, 92, 93, 97, 98, 102, 105, 108, 111, 112, 113, 114, 115, 116, 117, 118, 119, 121, 125, 141, 149, 150, 151, 152, 153, 155
zdrojový kód · 7, 11, 12, 13, 17, 22, 27, 30, 34, 43, 47, 50, 56, 57, 147
zprávy · 71, 78, 82, 87, 150

Ž

žádost/odpověď · 7, 8