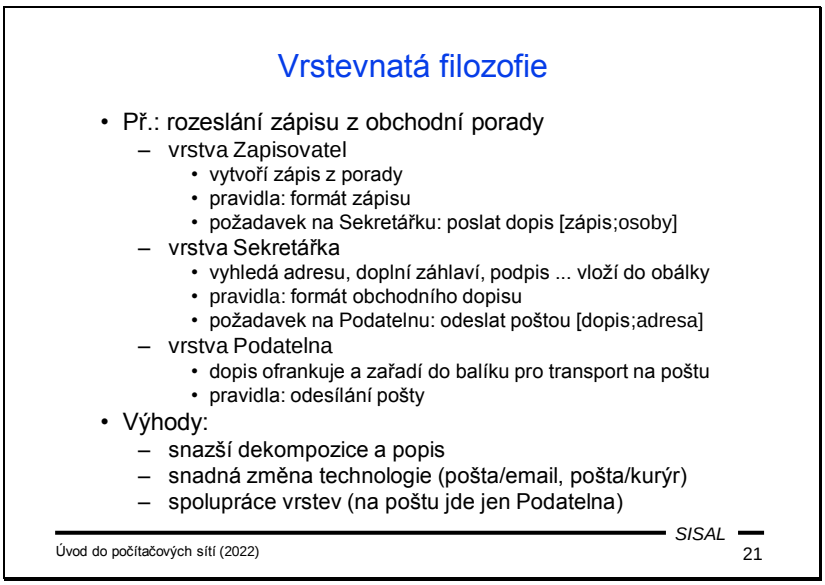
Snímek 21




K popisu síťové komunikace použijeme vrstevnatou filozofii. Výhody tohoto přístupu se pokusíme ukázat na příkladu z reálného života.

Představte si, že určitá společnost pořádá pravidelné schůzky a zápis z každého zasedání se zaznamená a zašle všem účastníkům.

Bez vrstevnatého přístupu musíme novému zaměstnanci popsat tento proces jako komplexní úkol, počínaje zapisováním, formátováním, řazením do obálek až po jejich odnesení na poštu.

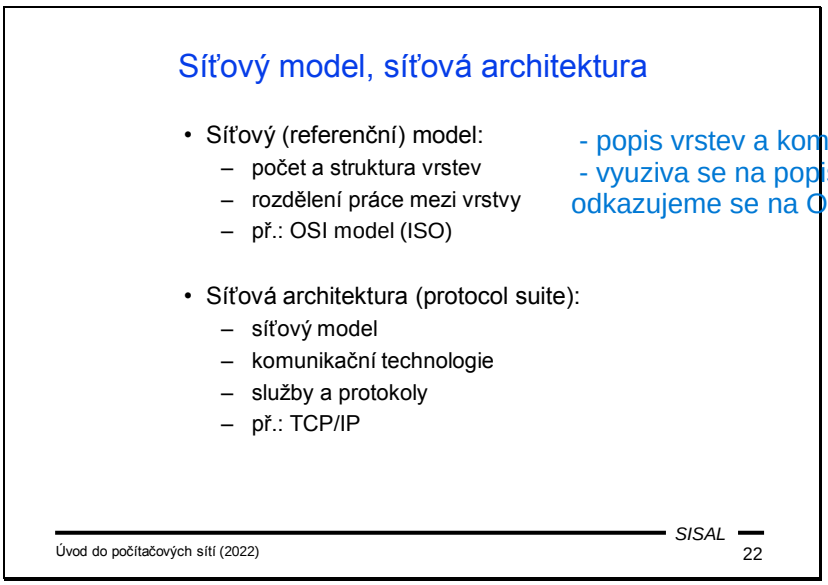
Jinou možností je rozdělit úkol do tří vrstev:
Vrstvu **Zapisovatel** představuje osoba, která pořizuje zápis. Musí dodržovat pravidla společnosti o tom, jak musí zápis vypadat – to lze považovat za *protokol*. Práce zapisovatele končí v okamžiku, kdy je text zápisu připraven a předává se další vrstvě, vrstvě Sekretářka.
Pro vrstvu **Sekretářka** není důležité, odkud a jaká data k ní přicházejí. Předávají se této vrstvě spolu s instrukcí, že má dopisy rozeslat na seznam adres účastníků schůzky – to lze považovat za rozhraní mezi vrstvami [*data=zápis*; *řídící data=seznam osob*]. Sekretářka převezme zápis, zformátuje ho dle pravidel pro obchodní dopis (záhlaví se správně umístěnou adresou, vlastní dopis a zápatí), vloží dopisy do obálek a napíše na ně adresy. Obálky jsou pak předány další vrstvě, Podatelně. Sekretářka v zásadě ani nemusí vědět, jaký druh přepravy bude použit.
Vrstva **Podatelna** přebírá obálky a vybírá způsob doručení podle aktuální strategie společnosti. Pokud společnost používá klasickou poštu, obálky se orazítkují, zabalí do balíčku a odnesou na poštu.

Jaké jsou výhody tohoto pracovního postupu?

- Celý proces je rozložen do jednoduchých úkolů, které lze snadno popsat novým zaměstnancům.
- Konkrétní realizaci jednoho jednoduchého úkolu lze snadno nahradit jiným řešením – pokud se společnost rozhodne přestat používat poštu a začít používat poslíčky, nemusí o této změně vědět žádná jiná vrstva kromě Podatelny.
- Úspora zdrojů díky spolupráci jednotlivých vrstev – není např. nutné, aby všechny sekretářky chodily na poštu.

V případě počítačové sítě získáváme s vrstevnatým přístupem naprosto stejné výhody – snadnost popisu, spolupráci mezi vrstvami a možnost změny technologie.

Snímek 22

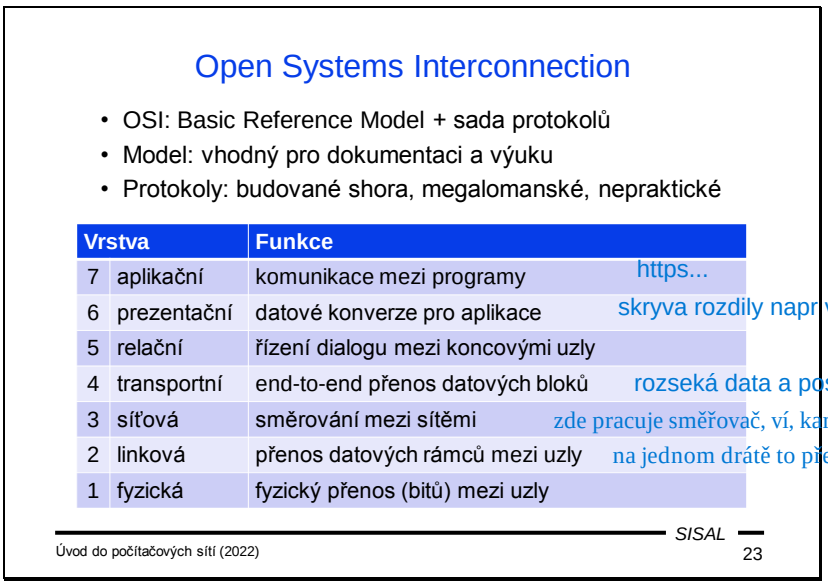



Když mluvíme o vrstevnaté struktuře počítačové sítě, rozlišujeme dva pojmy:

*Síťový model* popisuje počet vrstev, jejich strukturu a úkoly. Příkladem takového modelu je **model OSI** (Open Systems Interconnection) vytvořený Mezinárodní organizací pro normalizaci (ISO).

Vezmeme-li model a přidáme konkrétní služby, technologie, protokoly, rozhraní mezi vrstvami atd., získáme *síťovou architekturu*. Příkladem je sada protokolů **TCP/IP**.

Snímek 23




OSI model zavedla v roce 1978 Mezinárodní organizace pro normalizaci (ISO). Jedná se o síťovou architekturu sestávající ze základního referenčního modelu a souboru protokolů. Model byl sice navržen velmi dobře a až dosud se používá pro dokumentaci a výuku počítačových sítí, ale architektura jako celek byla výrazně méně úspěšná. Byla navržena shora, rozhodnutí odpovídala podmínkám v ideálním světě a byla příliš složitá na to, aby dosáhla jednoduchých cílů. Navíc byla poměrně nepružná a uživatelsky nepřívětivá…

Vrstva 1, **fyzická**, je zodpovědná za odesílání a příjem bitových dat mezi uzlem a fyzickým médiem. Neví nic o logice přenášených dat a zabývá se pouze fyzickými aspekty přenosu.

Vrstva 2, **linková**, zajišťuje přenos dat mezi dvěma přímo propojenými uzly. Jejím úkolem je určit zdroj a cíl přenosu, rozsah přenášených dat a odhalit a případně opravit chyby, které se mohly během fyzického přenosu vyskytnout.

Vrstva 3, **síťová**, zajišťuje přenos datových bloků s proměnlivou, ale omezenou délkou (pakety) mezi dvěma uzly, které se mohou nacházet v různých sítích. Pokud je zdrojový a cílový uzel ve stejné síti, síťová vrstva pouze předá požadavek na přenos linkové vrstvě. Pokud jsou v různých sítích, síťová vrstva je zodpovědná za nalezení cesty a směrování dat přes mezilehlé uzly.

Vrstva 4, **transportní**, je zodpovědná za přenos a příjem datových bloků s neomezenou délkou mezi zdrojovými a cílovými aplikacemi. Vrstva stále nerozumí sémantice dat, ale je schopna rozpoznat jejich příslušnost k určitému komunikačnímu kanálu mezi dvěma aplikacemi. Některé protokoly na této vrstvě umožňují *segmentaci* příliš velkých datových bloků a jejich opětovné sestavení na cílovém

uzlu. Některé protokoly poskytují *spolehlivý* přenos, tj. buďto zaručí úspěšný přenos celého bloku dat, anebo vrátí chybový stav.

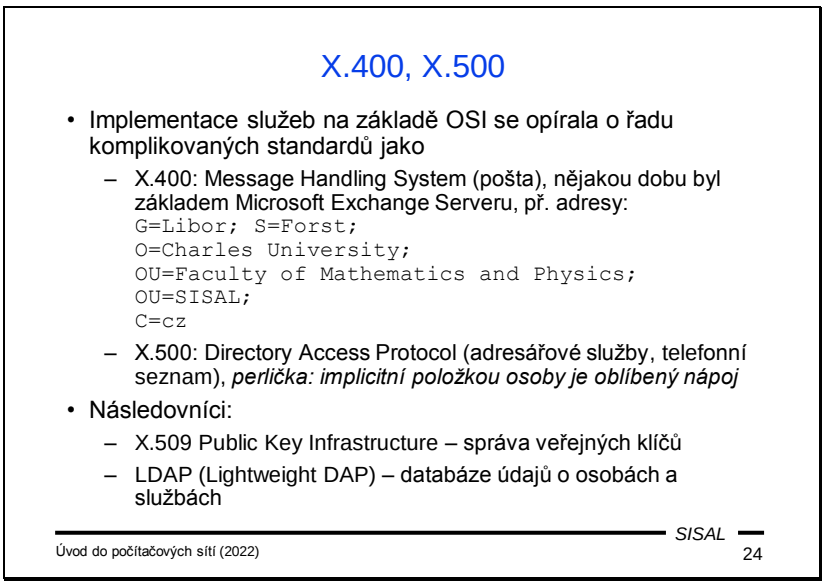
Vrstva 5, **relační**, řídí dialog mezi aplikacemi. V mnoha síťových modelech je jeho práce ve skutečnosti pokryta sousedními vrstvami.

Vrstva 6, **prezentační**, poskytuje prostředky pro skrytí rozdílů mezi implementacemi sémantiky dat na různých platformách, např. endianity celých čísel (hodnota je uložena do bajtů různými způsoby na různých počítačích), nebo konce textových řádků (LF=0x0A na systémech UNIX vs. CR+LF=0x0D0A na systémech Windows).

Vrstva 7, **aplikační**, je vrstva nejbližší koncovému uživateli. Aplikace zprostředkovává interakci mezi uživatelem a aplikačním protokolem, a ten implementuje komunikaci potřebnou k vykonání požadavku uživatele.

Pochopení funkce vrstev OSI modelu je důležité, protože při řešení různých otázek síťové komunikace musíme pečlivě zvážit, jakých vrstev se problém týká a které řešení lze použít na které vrstvě.

Snímek 24




Standardy OSI se označují podle schématu X.*číslo*. Pro příklad se můžeme podívat na dva nejdůležitější:

- **X.400 (Message Handling System)** byla implementace elektronické pošty v OSI. V určité době byl tento standard dokonce základem Microsoft Exchange Server. Zajímavostí je, že X.400 používal poměrně složitou strukturu adres. Tato struktura byla inspirována klasickými poštovními adresami - mají také mnoho komponent (jméno, ulici, číslo, PSČ, město, zemi) a jejich množství nikoho nepřekvapuje. Takové adresy mají ale jednu zásadní výhodu - jsou jednoznačné! V „počítačové éře“ se však nikomu nechce příliš psát a tak zvítězily sice nejednoznačné, ale mnohem kratší současné internetové emailové adresy.

- **X.500 (Directory Access Protocol)** byl adresářová služba, jedna z prvních implementací něčeho jako telefonní seznam nebo Zlaté stránky. Používá stejnou jednoznačnou strukturu identifikace osob jako X.400. Jako demonstraci toho, jak komplexní tato služba může být, se autoři rozhodli zahrnout oblíbený nápoj jako jeden z implicitních atributů osoby!

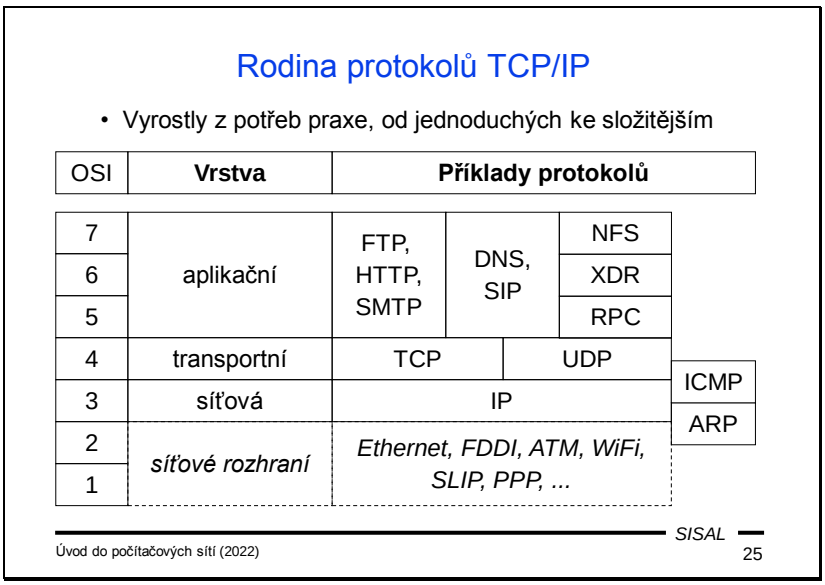
Přestože služby OSI nebyly široce implementovány, některé jejich nápady byly přínosné a zůstaly v některých moderních protokolech a datových strukturách:

- Asymetrická kryptografie potřebuje infrastrukturu pro správu veřejných klíčů. Každý veřejný klíč musí mít zcela jedinečný identifikátor. K tomu je ideální struktura adres navržených pro e-mailové a adresářové služby OSI.

- DAP v X.500 byl příliš složitý na to, aby mohl být implementován např. na osobních pracovních stanicích, přestože jeho základní myšlenka byla dobrá.

Řešením bylo vytvořit jednodušší verzi při zachování klíčových funkcí. Tento protokol se nazývá LDAP a v současné době je běžným řešením pro získávání a správu informací o uživatelích a službách.

Snímek 25




Na rozdíl od OSI byla rodina protokolů TCP/IP navržena odspodu pomocí menších stavebních bloků - protokolů, které řeší jeden kompaktní problém. Nicméně jak jsme předeslali, použijeme OSI model jako referenční model k popisu úloh řešených konkrétními vrstvami a protokoly TCP/IP.

Vrstvy OSI 1 a 2 nejsou součástí TCP/IP, které se na ně odkazuje jako na „síťové rozhraní“. Protokoly používané na těchto vrstvách odpovídají typu média využívaného pro přenos.

Nejnižší vrstva TCP/IP, která odpovídá vrstvě OSI 3, se nazývá **síťová vrstva**, stejně jako v OSI, a i její funkce je stejná. Protokol používaný na této vrstvě se nazývá *Internet Protocol* (IP). Ve skutečnosti v současné době existují dvě verze tohoto protokolu (IPv4 a IPv6), ale v tuto chvíli pro nás rozdíly mezi nimi nejsou důležité.

Další vrstva TCP/IP, **transportní vrstva**, má také stejný název a funkci jako v modelu OSI. V této vrstvě se v TCP/IP používá více protokolů, z nichž nejčastější jsou *Transmission Control Protocol* (TCP) a *User Datagram Protocol* (UDP). Rozdíly mezi nimi vysvětlíme na dalším slajdu.

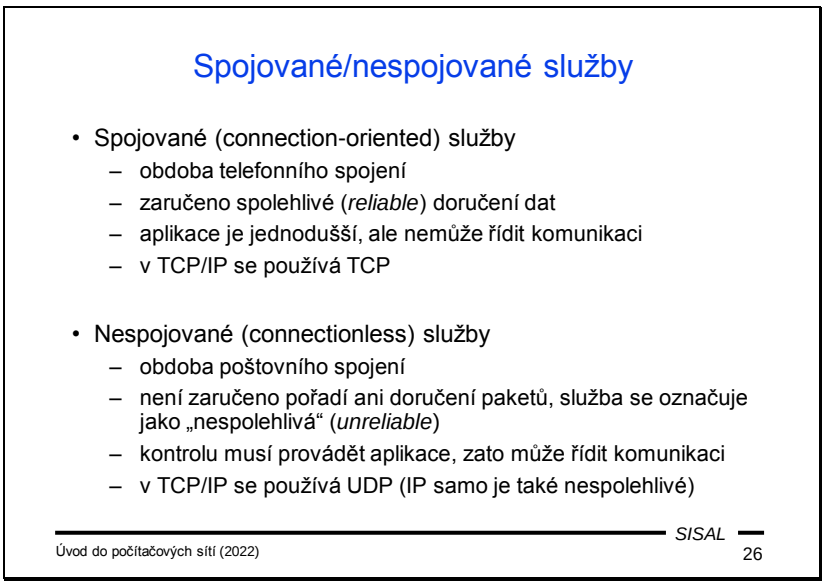
Funkce posledních tří vrstev OSI jsou v TCP/IP sloučeny do jedné vrstvy, **aplikační vrstvy**. Ukázalo se totiž, že pro většinu protokolů je snazší definovat pravidla dialogu a sémantiku dat přímo v aplikačním protokolu bez použití speciálních mezivrstev. Existuje několik výjimek, jako je Network File System, který používá speciální protokoly XDR (eXternal Data Representation) a RPC (Remote Procedure Call) na vrstvách 5 a 6. Autoři každého aplikačního protokolu si mohou vybrat, který protokol transportní vrstvy je pro aplikaci nejlepší, takže některé protokoly (např. FTP, HTTP,

SMTP) používají TCP, některé (např. NFS) používají UDP a některé (např. DNS, SIP) mohou používat oba.

Existuje také několik protokolů pro řízení TCP/IP, které stojí mimo přísnou hierarchii vrstev. Konkrétně zmíníme Address Resolution Protocol a Internet Control Message Protocol, o kterých si budeme podrobněji povídat později.

Poznamenejme, že samostatně používané termíny TCP a IP odkazují na konkrétní protokoly ve 4. a 3. vrstvě, zatímco pojem „TCP/IP“ označuje celou rodinu protokolů (což je poněkud nefér vůči opomenutému protokolu UDP).

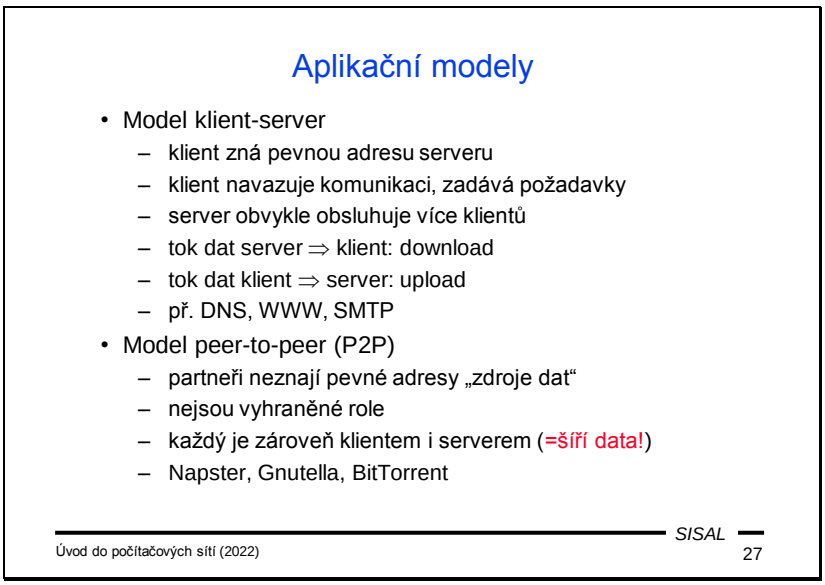
Snímek 26




**Transaction Control Protocol** (TCP) je určen pro spojované služby. Příkladem takové služby z běžného života je telefonní hovor. Když někomu zavoláte, telefonní operátor naváže spojení a vy se už nemusíte starat o to, zda budou vaše věty přeneseny všechny a ve správném pořadí. Aplikace (vy) tedy může být velmi jednoduchá a veškerá odpovědnost spočívá na spodní vrstvě. Stejný princip používá TCP. Poskytuje spolehlivé doručování paketů, takže aplikace zavolá pouze funkci „odešli datový blok na místo určení“ a čeká. TCP *segmentuje* data na menší bloky, odesílá je v jednotlivých paketech, potvrzuje jejich úspěšné doručení, v případě selhání je znovu odesílá a nakonec vrací řízení aplikaci s úspěšným nebo neúspěšným výsledkem. Implementace TCP je proto velmi složitá, zato implementace aplikace je snadnější. Nevýhodou tohoto přístupu je, že aplikace ztrácí kontrolu nad celým procesem, nemůže pružně reagovat na aktuální stav sítě. Prostě jen zavolá funkci a čeká.

**User Datagram Protocol** (UDP) je určen pro nespojované služby. Příkladem z běžného života je klasická pošta. Pokud odešlete tři zprávy během tří dnů, nikdy nevíte, zda budou doručeny všechny a ve správném pořadí. Aplikace (vy) proto musí buďto sama řešit potvrzování doručení a opětovné odeslání ztracených dat, nebo musí být vůči ztrátě dat odolná. Stejně se chová UDP. Je velmi jednoduchý, pouze přenáší jednotlivé pakety a odpovědnost za správnost přenosu ponechává na aplikaci. Výhodou je, že aplikace může okamžitě reagovat na aktuální situaci. Když dojde k nějakému problému, aplikace může zkusit jiný zdroj dat nebo se může uživatele dotázat, zda má přenos zastavit, apod.

Snímek 27




Jak jsme již zmínili, síťová komunikace umožnila vznik nových modelů aplikací, kdy na dálku spolupracují dvě části aplikace.

Mluvili jsme o modelu **klient-server**. Klient musí znát jednu nebo více adres, kde lze požadovanou službu nalézt, pokusí se ji kontaktovat a v případě úspěchu zahájí výměnu dat se serverem, aby vyhověl požadavkům uživatele. Role klienta a serveru jsou dobře definovány (z hlediska komunikace, ne nutně z hlediska toku dat - klient může klidně nahrávat data na server, tj. být zdrojem dat).

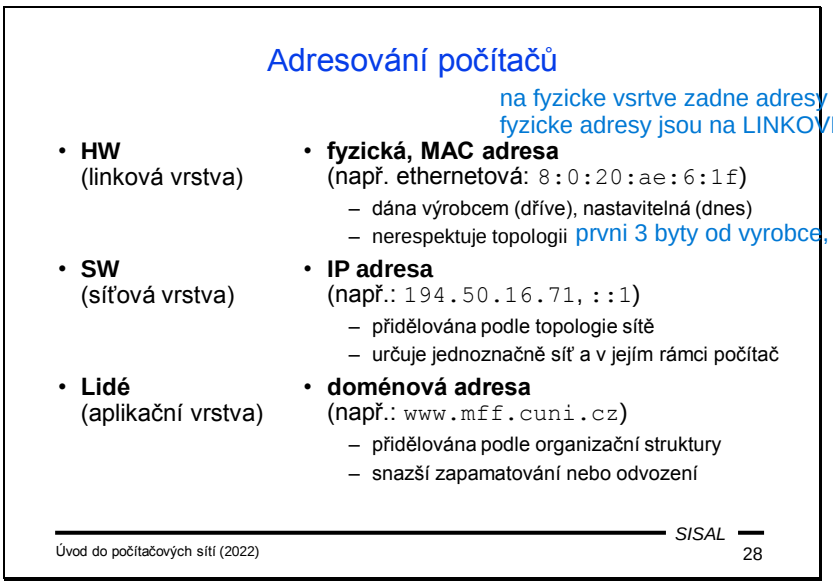
Důležitou skutečností je, že význam pojmů klient a server souvisí s konkrétní aplikací, nikoli s počítačem. Je zcela běžné, že počítač je serverem pro jeden typ komunikace, zatímco v jiném hraje roli klienta. Typická implementace serveru dokáže přijímat více příchozích požadavků od několika klientů paralelně.

Model klient-server však není jediným možným modelem aplikace. Existuje také takzvaný model **peer-to-peer**. Klíčovou vlastností tohoto modelu je, že uživatelská aplikace nepotřebuje znát adresu zdroje dat (i když často má nějaké počáteční adresy pro urychlení celého procesu). Aplikace je schopna si sama najít další počítače, na kterých stejná aplikace běží a je připravena k obousměrné výměně dat. Aplikace spuštěná na našem počítači totiž současně reaguje na všechny příchozí požadavky z jiných instancí aplikace a odesílá jim požadovaná data. Neexistují tedy žádné vyhraněné role, jako je klient a server.

P2P aplikace mají obecně špatnou pověst, protože se často používají k distribuci dat, k jejichž distribuci nemá uživatel právo. Problém není v samotných aplikacích, ale v datech. Pokud vaše aplikace stahuje data nelegálně, nedopustili jste se žádného provinění, ale jakmile začne data nabízet ostatním, stáváte se tím, kdo nelegálně šíří

data, a to už problém je. Jedinou možností je zakázat aplikaci, aby sama poskytovala data, pokud to je v dané aplikaci možné.

Snímek 28




Když chce klient kontaktovat server, potřebuje znát jeho adresu. V závislosti na vrstvě, ve které probíhá komunikace, se používají různé druhy adres.

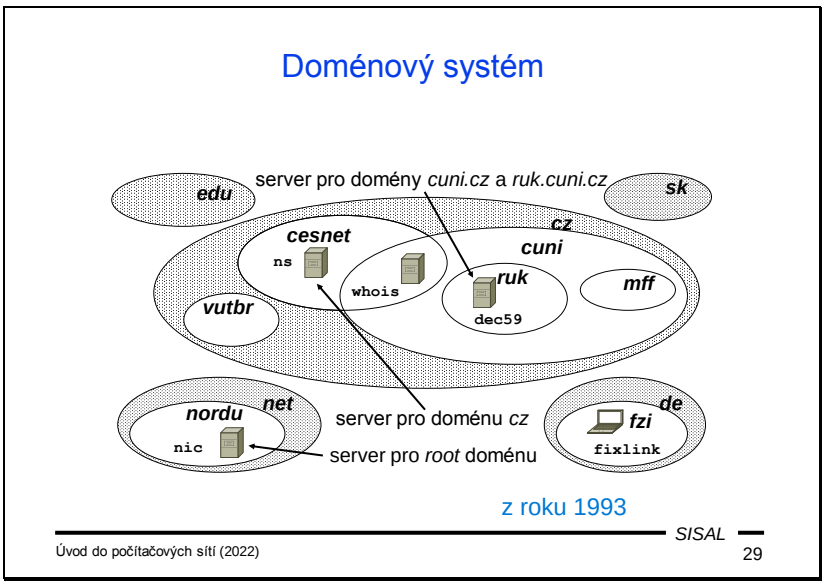
Na linkové vrstvě se používají **MAC adresy** (Media Access Control). Protože jsou poměrně úzce propojeny s hardwarem, někdy se jim říká „fyzická“ adresa, přestože s fyzickou vrstvou OSI nemají nic společného! Příkladem MAC adres jsou ethernetové adresy. Jsou šestibajtové a skládají se z prefixu výrobce (3 bajty) a čísla síťové karty (3 bajty). Původně bývaly adresy vypálené do karty, takže bylo možné efektivně omezit přístup k síťovým prostředkům pomocí MAC adresy odesílatele. Současné síťové karty však mají adresu MAC uloženou v paměti, a proto ji lze snadno zfalšovat. Vzhledem ke své povaze nelze tyto adresy použít pro komunikaci mezi různými sítěmi, protože nerespektují žádnou konkrétní topologii sítě.

Při komunikaci mezi sítěmi tedy musíme použít takový typ adresy, který respektuje topologii sítě. V TCP/IP se používají **IP adresy**. Už jsme zmínili, že existují dvě verze protokolu, takže máme dva formáty IP adres. Každému počítači je přiřazena adresa podle toho, kde je připojen k síti. Každá adresa má dvě části – adresu sítě (používá se pro směrování mezi sítěmi) a adresu počítače (používá se uvnitř sítě). Hranice mezi těmito dvěma částmi IP adresy závisí na konkrétním nastavení sítě.

IP adresy jsou vhodné pro správné fungování síťové komunikace, ale jsou poněkud uživatelsky nepřívětivé. V běžné praxi proto používáme ještě jiný formát adres, **doménové adresy**. Místo čísel používají tyto adresy mnemonický formát, *doménová jména*, respektující organizační strukturu. Proto je mnohem snazší si tyto adresy zapamatovat nebo odvodit. Hierarchie je u těchto adres zprava doleva: poslední jméno je takzvaná *doména nejvyšší úrovně* (TLD), např. kód země, zatímco název zcela vlevo je obvykle název hostitele nebo služby.

K převodu mezi doménovými jmény a adresami IP se používá systém DNS (Domain Name System). K převodu mezi síťovými a MAC adresami se používá protokol ARP (Address Resolution Protocol).

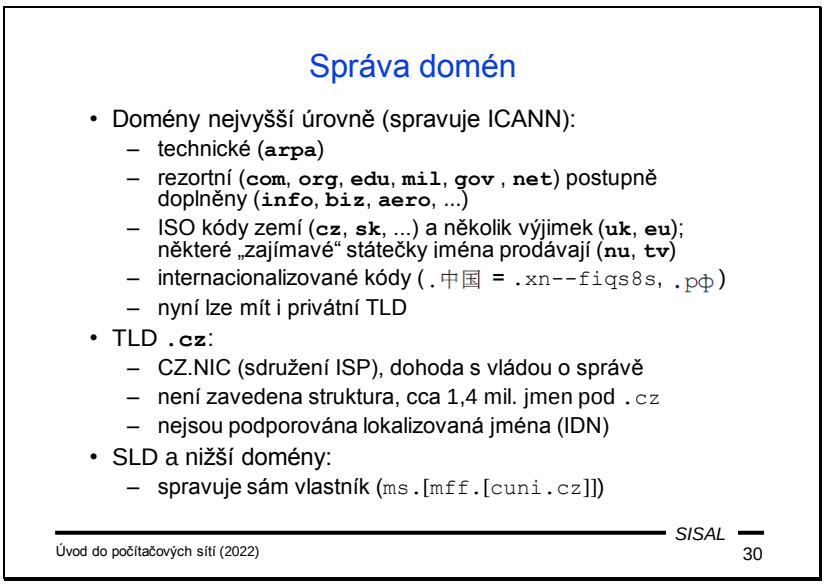
Snímek 29




Systém doménových jmen (Domain Name System) je hierarchická struktura *zón*, jež obsahují informace o podřízených počítačích a zónách. Tyto informace jsou uloženy v databázi sdílené množinou jmenných serverů (*nameserverů*) pro danou zónu a tyto servery poskytují klientům odpovědi na jejich dotazy pomocí protokolu DNS.

Každý počítač by měl mít přiřazeno doménové jméno, které odpovídá adresám, jež aktuálně používá. Počítač ovšem může mít více jmen, zvláště pokud na něm běží více služeb pro více domén.

Snímek 30

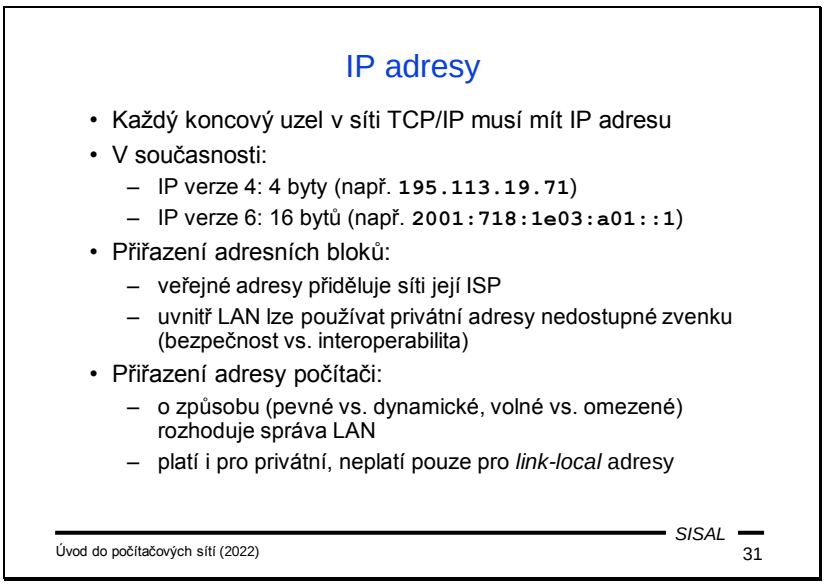



Domény nejvyšší úrovně (TLD) spravuje ICANN (Internet Corporation for Assigned Names and Numbers). Struktura systému doménových jmen byla definována v RFC 920 a podmínky pro TLD byly původně relativně přísné. Existuje doména **arpa**, původně zamýšlená jako dočasná pro staré počítače ARPANETu, později změněná na doménu používanou pro technické účely. Druhou skupinou domén byly domény pro jednotlivé státy pojmenované podle dvouznakových ISO kódů zemí (s některými výjimkami, jako je **uk** nebo **eu**). V USA pak bylo pět „rezortních“ (**com** pro komerční společnosti, **org** pro nekomerční organizace, **edu** pro vzdělávací a výzkumné instituce, **mil** pro armádu, **gov** pro vládu a později také **net** pro síťové organizace). Časem došlo logicky ke konfliktům názvů, zejména v doméně **com**. Proto byly později přidány některé nové TLD (**info**, **biz**, **aero**, ...), ale to samozřejmě nemohlo problém vyřešit. ICANN pod velkým tlakem nakonec zcela otevřela prostor jmen a v dnešní době může o individuální TLD požádat i soukromý subjekt.

Doménu **cz** spravuje CZ.NIC, korporace českých ISP. Doména nemá žádnou „rezortní“ strukturu kategorií jako v některých jiných zemích (Velká Británie, Rakousko), a proto přímo pod ní existuje přes milion jmen definovaných jako domény druhé úrovně (SLD). Přestože je v současné době technicky možné povolit lokalizovaná jména, CZ.NIC to naštěstí zatím nepovolil.

Domény druhé úrovně spravují jejich majitelé, kteří mohou rozhodovat o jejich další struktuře. Na UK jsme se rozhodli zachovat co nejvíce hierarchii a pod SLD **cuni.cz** jsou povoleny pouze domény pro fakulty a subjekty celouniverzitního charakteru. Podobně je organizována i doména naší fakulty, má strukturu podle kateder a budov (např. název domény **ms** je zkratkou lokality „Malá Strana“).

Snímek 31



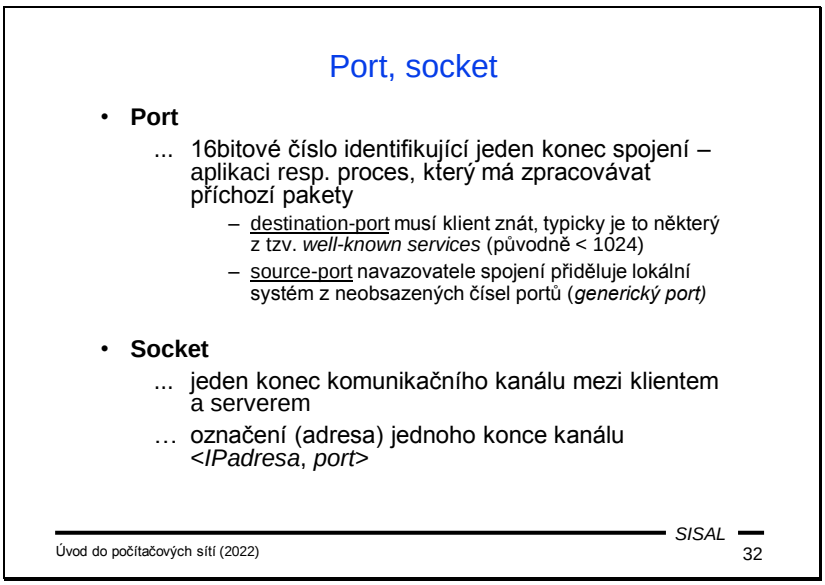

Každý koncový uzel TCP/IP sítě musí mít přiřazenou IP adresu. Jak již bylo zmíněno, v současné době používáme dvě verze protokolu a tím i adres. V České republice stále dominantní verze 4, která má adresy dlouhé 4 bajty a běžnou formou jejich zápisu je dekadická tečková notace. Novější verze 6 zvětšila rozsah adresy na 16 bajtů a běžně používaný formát zápisu je hexadecimální notace po dvoubajtových blocích oddělených dvojtečkami. Vzhledem k velkému počtu nul u současných adres lze nejdelší skupinu nulových bloků v adrese nahradit zápisem „::“.

Každá adresa má prefix definující síť, do níž adresa patří. Přiřazení těchto tzv. *síťových adres* musí splňovat určité podmínky:

- Bloky **veřejných** adres, které mohou přímo komunikovat se zbytkem světa mimo LAN, přiděluje ISP (nebo jiný orgán), který připojuje síť k Internetu.
- Uvnitř LAN se správce sítě může rozhodnout použít jeden nebo více bloků tzv. **privátních** adres. Ty jsou nepřístupné z Internetu (což zvyšuje bezpečnost, ale komplikuje interoperabilitu), a aby mohly lokální počítače přistupovat ke zdrojům na internetu, musí směrovač na hranici LAN používat překlad adres (Network Address Translation).

Způsob přiřazování adres počítačům v místní síti zvolí správa sítě a všechny počítače musí tato pravidla dodržovat. Přiřazení může být statické (každý uzel má předdefinovanou IP adresu) nebo dynamické (adresa je přiřazena na vyžádání), a rovněž volné (může se připojit jakýkoli hostitel) nebo omezené (pro připojení se musí hostitelé autentizovat). Tato pravidla platí i pro **privátní** adresy. Jedinou výjimkou jsou takzvané *link-local* adresy – ty si volí každý počítač sám (což přináší riziko kolizí duplicitních adres) a umožňují mu komunikaci pouze v rámci přímo připojené sítě (a pouze v případě, že link-local adresy používají i jiné počítače v síti).

Snímek 32




Znalost adresy cílového počítače není pro zahájení komunikace postačující. Musíme totiž ještě specifikovat konkrétní službu, protože na vzdáleném stroji může běžet více serverů. A ze stejného důvodu musíme také umět nějak identifikovat „náš konec“ komunikačního kanálu, protože když přijde od serveru odpověď, musí být jasné, jaké aplikaci má být doručena – počítač musí například umět vložit příchozí obrázek do správného okna webového prohlížeče …

Termín **socket** označuje jeden konec komunikačního kanálu mezi klientem a serverem. Soket je identifikován IP adresou a 16bitovým celočíselným číslem nazývaným **port**, které slouží k rozlišení různých služeb či aplikací.

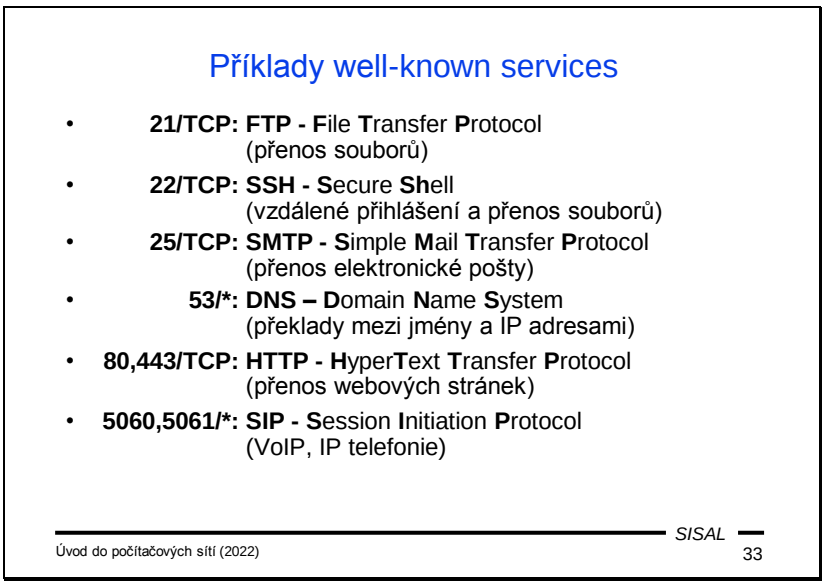
Klienti ovšem neznají situaci na cílovém počítači, takže aby bylo možné správně vyplnit *číslo cílového portu*, používají běžné servery předdefinovaná čísla portů, tzv. **well-known services**. Například webové servery používají implicitně číslo portu 80.

Well-known services však nejsou řešením pro stanovení *čísla zdrojového portu*, protože na klientském počítači potřebujeme identifikovat více klientů stejné služby. Obvyklé řešení tedy je, že v okamžiku, kdy klientský software připravuje soket pro připojení k serveru, požádá operační systém o přiřazení nějakého nepoužitého generického čísla portu (mimo rozsah well-known services). Klient používá tento port pro celou komunikaci se serverem a po ukončení spojení port uvolní, takže operační systém ho má k dispozici pro jiné aplikace.

Z matematického hlediska je jednoznačnou identifikací spojení v TCP/IP uspořádaná pětice <zdrojová IP adresa, zdrojový port, cílová IP adresa, cílový port, typ protokolu transportní vrstvy>. Z toho vyplývají dvě pravidla:

- Dva různé kanály stejné aplikace mezi dvěma počítači se musí lišit alespoň ve zdrojovém portu.
- Stejná čísla portů lze použít pro dva různé komunikační kanály, pokud nepoužívají stejný protokol transportní vrstvy.

Snímek 33

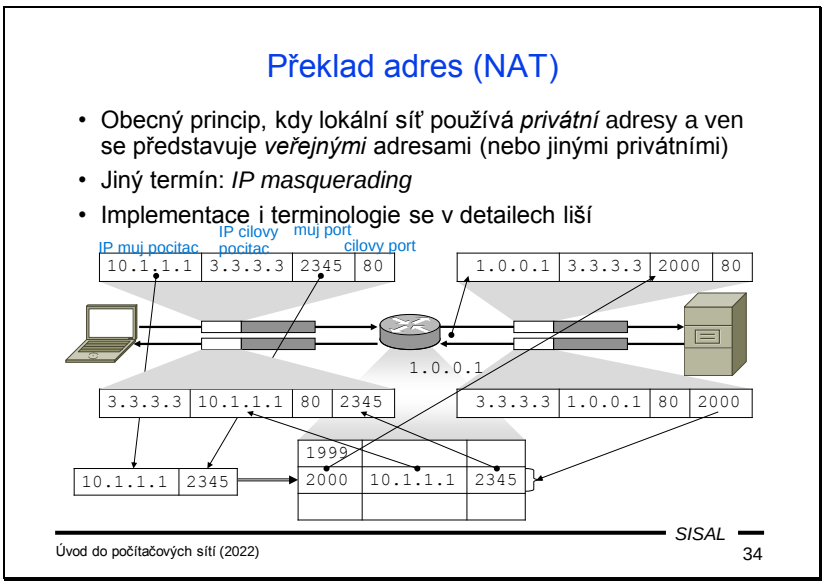



Příklady čísel portů na tomto slajdu patří k základní síťové gramotnosti, snad kromě posledního bodu, který pouze dokumentuje, že některé známé služby dnes nespadají do rozsahu od 1 do 1023...

Čísla portů pro danou službu jsou obvykle vyhrazena pro oba protokoly TCP a UDP, a to i když konkrétní aplikační protokol používá jen jeden z těchto transportních protokolů.

HTTP používá port 80 pro nešifrovaná a port 443 pro šifrovaná spojení. SIP používá různé porty pro různé typy zpráv.

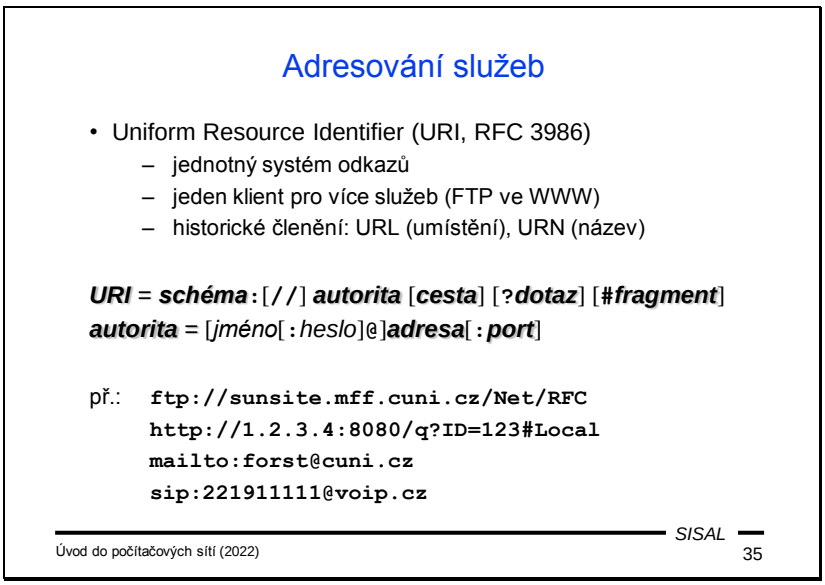
Snímek 34




Koncept portů hraje velmi důležitou roli při překladu adres (NAT). Počítače s privátní adresou nemohou komunikovat přímo se světem mimo místní síť, protože externí počítač nemá tušení, jak směrovat pakety při odesílání odpovědi.

Základní koncepce NAT je založena na tom, že první paket směřující od klienta k serveru je zachycen směrovačem na perimetru LAN. Směrovač, který má zapnutou funkci LAN, upraví obsah paketu tak, aby externí hostitel mohl korektně odpovědět. Směrovač si proto uloží socketovou adresu (IP adresu a port) příchozího požadavku a nahradí v paketu patřičná pole vlastní externí IP adresou a nějakým portem, který je na směrovači volný. Server tedy posléze odesílá odpověď na tuto upravenou socketovou adresu. Když odpověď dorazí zpátky na směrovač, ten vyhledá příslušnou původní adresu socketu (podle cílového portu použitého serverem), změní patřičná pole v odpovědi zpět na původní hodnoty (IP adresu a port z požadavku) a doručí odpověď klientovi.

Snímek 35




Aby nějaká aplikace mohla navázat spojení ke konkrétnímu zdroji dat, musí pracovat s komplexní informací o požadované službě. Situace před vznikem WWW byla poněkud nepřehledná a existovala řada různých aplikací pro různé služby. Webový prohlížeč byla první aplikace, která používala různé protokoly (FTP, Gopher, WWW) a odkazy v HTML musely pokrýt různé služby. Proto bylo třeba zavést obecný formát odkazu, který kombinuje metodu přístupu ke zdroji, adresu serveru, přihlašovací údaje uživatele, detailní umístění zdroje atd. Původní myšlenkou bylo definovat odkaz pomocí tzv. URI (Uniform Resource Identifier), který může obsahovat buď konkrétní umístění zdroje (URL, Uniform Resource Locator), anebo jen název služby bez explicitního umístění (URN, Uniform Resource Name) . Odkazy URN však nikdy nebyly úspěšně implementovány, takže termíny URL a URI jsou víceméně zaměnitelné.

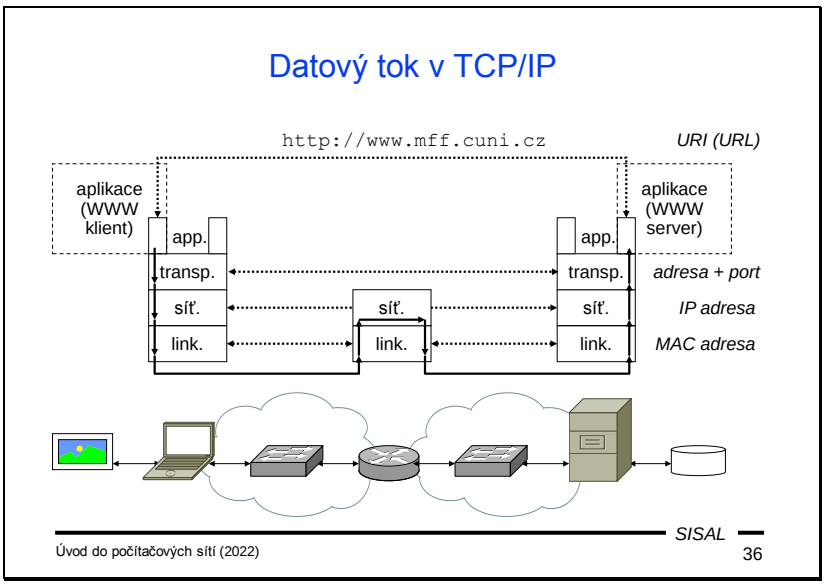
První část adresy URI je **schéma** (následované dvojtečkou). Častou chybou je označování této části jako „protokol“, k čemuž svádí to, že mnoho schémat se shoduje s názvem protokolu, který by měl být použit pro přístupu ke zdroji. Každá adresa URI musí obsahovat schéma, pokud není zřejmé z kontextu - např. „http“ je implicitní schéma odkazu na stránce HTML, takže tam ho psát nemusíme, v jiném kontextu může ale vynechané schéma znamenat něco jiného.

Další částí adresy URI je **autorita**, u některých typů URI prefixovaná dvojicí lomítek. Autorita určuje server (nebo doménu) společně s informacemi o uživateli (je-li to nutné). Pokud server používá nějaký nestandardní port, je třeba jej zde uvést, jinak aplikace použije defaultní well-known port. U odkazů na HTML stránce lze autoritu vynechat, pokud je stejná.

Mnoho typů URI obsahuje dále **cestu** k požadovanému zdroji v rámci serveru. Formát cesty je podobný cestám v souborových systémech (přičemž se jako oddělovač adresářů používá normální lomítko) a cesta často ve skutečnosti opravdu odpovídá skutečnému umístění stránky na souborovém systému serveru. Pokud je u odkazů na HTML stránce vynechána autorita, lze použít relativní cestu vzhledem k adresáři, kde stránka leží, podobně jako se relativní cesta používá v souborovém systému.

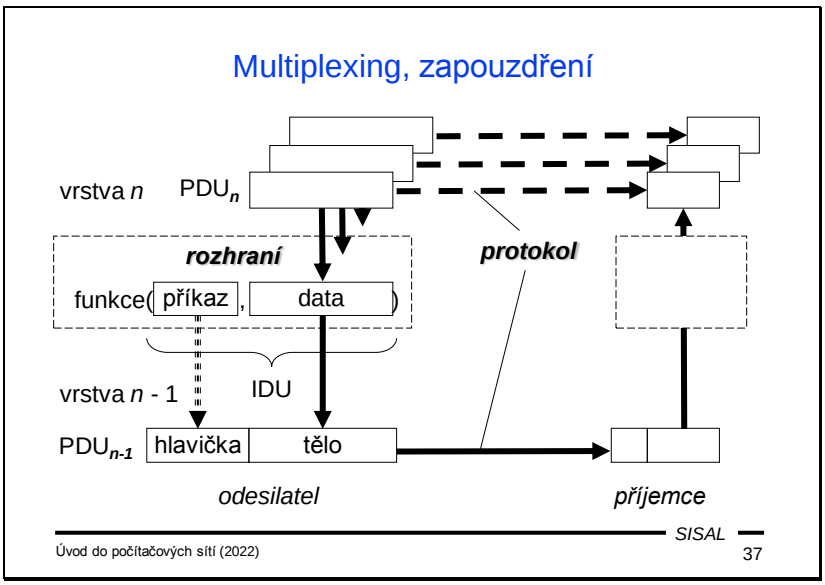
Některé typy URI mohou obsahovat další parametry používané k přesnější specifikaci požadavku; např. HTTP URI může obsahovat **dotaz** (a v něm např. data převzatá z HTML formuláře) nebo **fragment** (identifikátor bodu uvnitř stránky, na který se má prohlížeč přesunout).

Snímek 36




- Na aplikační vrstvě klient obvykle adresuje server pomocí URL. Aplikační vrstva předává data spolu s cílovou adresou a portem transportní vrstvě.
- Oba konce logického komunikačního kanálu na transportní vrstvě jsou identifikovány socketovými adresami (tj. IP adresou a číslem portu klienta a serveru). Transportní vrstva předává data spolu s cílovou IP adresou síťové vrstvě.
- Na síťové vrstvě je komunikační kanál definován IP adresami klienta a serveru. Klient se musí před odesláním dat rozhodnout, zda cílová adresa patří do stejné sítě. Podle tohoto rozhodnutí se zvolí tzv. *next-hop* uzel a data plus next-hop adresa jsou předány linkové vrstvě.
    - Pokud je server ve stejné síti jako klient, next-hop uzlem je cílový server.
    - Jinak musí být nalezen router schopný přesměrovat pakety do cílové sítě a tento router je next-hop uzlem.
- Na linkové vrstvě komunikační kanál propojuje uzel s next-hop uzlem (tj. ne nutně s cílovým počítačem) takže cílová adresa na vrstvě datového spojení je MAC adresa next-hop uzlu.
- Jakmile jsou data doručena prostřednictvím fyzické vrstvy na konec komunikačního kanálu linkové vrstvy, liší se chování podle povahy uzlu:
    - Pokud je uzlem cílový server, jsou data předána k dalšímu zpracování vyššími vrstvami.
    - Pokud je uzlem směrovač, jsou data předána pouze síťové vrstvě a tato vrstva znovu rozhodne, který next-hop uzel musí být použit pro další doručení. Data s adresou next-hop uzlu jsou poté zase předána zpět linkové vrstvě a celý proces se opakuje.

Snímek 37




Důležitou metodou pro efektivní využití sítě je **multiplexing**. Termín označuje fakt, že několik komunikačních kanálů v určité vrstvě používá stejný komunikační kanál v podřízené vrstvě.

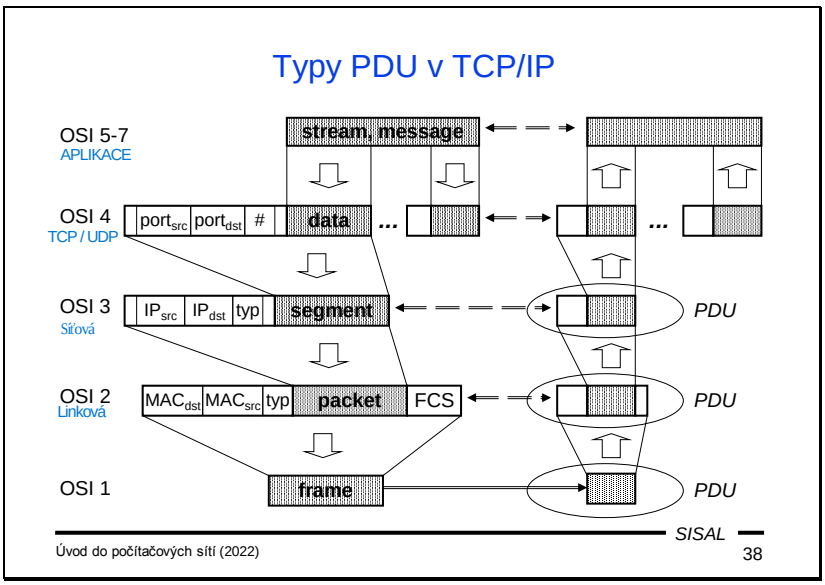
Obě strany komunikačního kanálu na libovolné vrstvě musí dodržovat sadu pravidel správného chování, tzv. **protokol**. Na jedné vrstvě může probíhat komunikace ve více protokolech současně.

Představme si, že na vrstvě *n* máme protokol, který definuje formát jednotek dat (PDU, **Protocol Data Unit**) této vrstvy. Software na vrstvě *n* dokončil svou práci a rozhodl se zavolat vrstvu *n*-1, aby realizovala přenos. Vezme jednotku dat ($PDU_n$) a přidá řídicí informace (jako je cíl, velikost, různé parametry doručení apod.) a zavolá službu nižší vrstvy. Obvykle má takové volání formu knihovní funkce, jejímiž parametry jsou data a řídicí informace ve formě určitého příkazu. Knihovní funkce vrací informaci o úspěchu nebo selhání vrstvy *n*-1. Tato výměna dat mezi vrstvami se nazývá **rozhraní** (interface) a jejímu formátu se říká **Interface Data Unit** (IDU).

Vrstva *n*-1 převezme řídicí informace, zpracuje je a poté připraví jednotku dat podle konkrétního protokolu této vrstvy. Tato PDU obvykle sestává z *těla* obsahujícího $PDU_n$ a *záhlaví* s řídicími informacemi. Tuto metodu nazýváme **zapouzdření** (encapsulation) dat vrstvy *n* do dat na vrstvě *n*-1. Nejběžnějším případem zapouzdření je právě $n \rightarrow n$-1, ale obecně je možné zapouzdřit PDU jakékoli vrstvy do jakékoli vrstvy, dokonce včetně stejné vrstvy nebo vyšších vrstev. Pokud například potřebujeme odeslat paket IPv6 přes síť IPv4, musíme $PDU_3$ zapouzdřit do jiného $PDU_3$.

Záhlaví musí mj. obsahovat identifikátor vrstvy *n*, protože na straně příjemce musí být provedeno **rozbalení** (decapsulation) a **demultiplexování**. Vrstva *n*-1 zkontroluje správnost řídicích informací a poté předá data (tělo) příslušnému softwaru na vrstvě *n*.

Snímek 38




Podívejme se, jak funguje multiplexing a zapouzdření v rodině protokolů TCP/IP.

Aplikační vrstva odesílá buď jednotlivé **zprávy** (u nespojovaných aplikací), nebo **proud** dat (stream, u spojovaných aplikací). Příslušnému protokolu na transportní vrstvě předává data a cílovou adresu soketu.

Pokud aplikace využívá protokol TCP, patřičný software převezme blok dat ze streamu, rozdělí (**segmentuje**) ho na menší bloky a připraví záhlaví obsahující mj. čísla zdrojového a cílového portu a „offset“ dat v rámci proudu, aby bylo možné na cílovém počítači data seřadit a znovu sestavit do bloků. UDP má mnohem jednodušší práci, pouze před každou zprávu připíše záhlaví s čísly portů. $PDU_4$ je poté předán síťové vrstvě spolu s cílovou IP adresou.

Síťová vrstva vezme segment, přidá své záhlaví se zdrojovou a cílovou IP adresou a číslem protokolu transportní vrstvy a vytvoří **paket**. Poté zkontroluje cílovou adresu, zda lze paket doručit přímo, anebo zda musí být použit next-hop router. Paket spolu s cílovou adresou linkové vrstvy je předán software linkové vrstvy.

*Varování*: termín „paket“ je třeba používat obezřetně a pečlivě rozlišovat, zda jde o jednotku dat třetí vrstvy OSI, anebo poněkud vágní termín používaný pro „přepínání paketů“.

Linková vrstva vezme paket a přidá své záhlaví s cílovou a zdrojovou MAC adresou a číslem protokolu síťové vrstvy. Protože se jedná o poslední „softwarovou“ vrstvu před tím, než jsou data předána „hardwaru“ (fyzické vrstvě), potřebujeme přidat rovněž nějakou metodu, jak zkontrolovat, zda byla data přenesena správně. PDU

linkové vrstvy (nazývaná **rámec**, frame) proto obsahuje také zápatí s tzv. Frame Check Sequence (FCS). Tato hodnota se vypočítává z obsahu zbytku rámce.

Posledním krokem je předání rámce fyzické vrstvě, která data přenese.

Po doručení do cílového uzlu fyzická vrstva data dekóduje a předá linkové vrstvě.

Linková vrstva přepočítá hodnotu FCS a zkontroluje, zda se rovná hodnotě uložené v rámci. Také zkontroluje cílovou MAC adresu, aby zjistila, zda rámec patří konkrétnímu uzlu. Podle čísla protokolu síťové vrstvy uloženého v záhlaví se poté rozhodne, jakému software na síťové vrstvě bude rozbalený paket předán.

Síťová vrstva zkontroluje cílovou IP adresu a předá rozbalený segment příslušnému software transportní vrstvy.

Rozsah práce transportní vrstvy závisí na použitém protokolu. V případě UDP je zpráva jednoduše doručena aplikaci. V případě TCP je segment pouze uložen a datový blok bude předán aplikaci až po přijetí všech segmentů.

Snímek 39

## Souhrn 2

- Popište rozdíl mezi TCP, UDP a IP.
- Jaké adresy se používají na aplikační, transportní, síťové, linkové a fyzické vrstvě?
- Jaký je rozdíl mezi klient-server a peer-to-peer modelem?
- Jak jsou spravována doménová jména?
- Jak se přidělují IP adresy?
- Jaký je smysl a princip NAT?
- Co je multiplexing a zapouzdření?

aplikační : doménové, IP - zapouzdření do URL (protokol...)
transportní : povídání dvou bodů (IP + port)
síťová : IP
linková : MAC
fyzicka : nic, fyzika vole

klient zná adresu serveru, naváže spojení/pošle zprávu, server odpovída

p2p - posílám všesměrovou zprávu, žádná vyhraněná role klient/server

hierarchicky
- top-level - centrální správa
- postupně níž a níž

obdrží správa sítě, uvnitř lokální sítě používají privátní adresy , privátní adresa se pak musí přeložit

NETWORK ADDRESS TRANSLATION - socketová adresa se na routeru přepíše, odešle se svojí adresou, pak obdrží odpověď, podle tabulky zpátky přeloží a pošle zpět

mnoho různých protokolů se pomocí zapouzdřování se posílají o level níž, na konci zase decapsulation.... IDKKKK